

GOLDEN FLUTES & GREAT ESCAPES

How to Write Adventure Games

For the **64**TM
Commodore[®]



Delton T. Horn



dilithium Press

**Golden Flutes and
Great Escapes
How to Write
Adventure Games
For the Commodore 64**

Golden Flutes and Great Escapes

**How to Write
Adventure Games
For the Commodore 64**

Delton T. Horn



**dilithium Press
Beaverton, Oregon**

© 1984 by dilithium Press. All rights reserved.

No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system without permission in writing from the publisher, with the following exceptions: any material may be copied or transcribed for the nonprofit use of the purchaser, and material (not to exceed 300 words and one figure) may be quoted in published reviews of this book.

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging in Publication Data

Horn, Delton T.

Golden flutes and great escapes: how to write adventure games for the Commodore 64.

Includes index.

1. Computer games. 2. Commodore 64 (Computer) — Programming.

3. Basic (Computer program language)

I. Title.

GV1469.2.H673 1984 794.8'2 84-12727

ISBN 0-88056-051-7 (pbk.)

Cover: Vernon G. Groff

Commodore 64 is a registered trademark of Commodore Computer Systems.

dilithium Press is a trademark of dilithium Press, Limited.

Printed in the United States of America

dilithium Press

8285 S.W. Nimbus

Suite 151

Beaverton, Oregon 97005

Table of Contents

Chapter 1–DISCOVERING ADVENTURE GAMES	1
Getting Started	3
Chapter 2–CREATING A PLOT	5
The Game Format	6
Naming Objects	8
Naming the Monsters	9
Natural Obstacles	10
Summary	10
Chapter 3–BEGINNING THE PROGRAM	11
Initialization	11
Introduction and Setting Values	14
Beginning the Main Play Routine	21
Adding Commands	27
Finding the Objects	37
Inventory	50
Blast Off	52
The EAT and DRINK Commands	53
Summary	57
Chapter 4–COMPLICATING THE GAME	59
Mapping Monsters	59
Squeanly Serpent	62
Ghost	63
Brinchley Beast	66
KuFu	67
Grimph	69

	Purofolee	70
	River	71
	Mountain/Ravine	72
	Marsquake	73
	Storm	75
	Funny-Colored Sky	77
	Dead Monsters	80
	Moving Past Monsters	81
	Moving Past Rivers and Mountains	84
	Expanding the CRY Command	84
	Expanding the EAT Command	86
	Expanding the DRINK Command	90
	Adding the FILL Command	92
	Adding INFLATE Command	95
	Adding CLIMB Command	96
	Open Box	100
	Expanding PRAY Command	102
	KILL Command	104
	TOUCH Command	114
	Summary	116
Chapter 5	– WRITING THE INSTRUCTIONS	119
	Purpose	119
Chapter 6	– THE COMPLETE “MARS” PROGRAM	125
Chapter 7	– GRAPHICS, SOUND, AND OTHER EXTRAS	157
	Graphics	158
	Applications for Graphics	160
	Sound Effects	161
	Real-Time Inputs	162
	SAVE GAME	163
	Summary	163
Chapter 8	– TREASURE HUNT	165
	Programming	175
	The Play	183
	Encountering Obstacles	185
	Possible Variations	186
	Summary	187

Chapter 9–THE GOLDEN FLUTE	189
Purpose	189
Characters	216
The Play	217
Playing the Game	221
Some Secrets of the Game	221
The Monsters	222
Expanding the Game	224
Chapter 10–THE GREAT ESCAPE	227
Gold Coins and Scoring	227
Obstacles	228
Batteries and Map	259
Aids for the Player	259
Summary	261
Chapter 11–MARKETING YOUR SOFTWARE	263
LOADING INSTRUCTIONS	267
INDEX	271

An Important Note

The publisher and the authors have made every effort to ensure that the computer programs and programming information in this publication are accurate and complete. However, this publication is prepared for general readership, and neither the publisher nor the authors have any knowledge about or ability to control any third party's use of the programs and programming information. There is no warranty or representation by either the publisher or the authors that the programs or programming information in this book will enable the reader or user to achieve any particular result.

Preface

Adventure games add a playful dimension to owning a home computer. This book gives examples of action-filled video games that bring you hours of fun; soon you will be writing your own original games for fun—and profit.

These games were written on a Commodore 64 microcomputer but can be easily adapted to any comparable home computer.

While you need not be a programming whiz to understand and enjoy these programs, beginners will want to become familiar with BASIC programming.

Chapter 1

Discovering Adventure Games

You move cautiously down a long, dark corridor beneath the castle of Nembuzur. You notice that your torch is beginning to burn a little low. You'll have to find a new one soon.

At the end of the hall, you come to three closed doors. After a moment's hesitation, you open the third door. A saber-toothed tiger leaps out at you and injures your left arm. Quickly you draw your magic sword. . . .

Is this a dream? No, you are a character in a video adventure game, a rapidly growing American pastime. While so much of our entertainment these days is passive—television, movies, books, spectator sports—the more active video games have caught on in a big way. Most of these games are skill-oriented. They test your reflexes but not your creativity and imagination. You either shoot down the aliens, or they shoot you down. There are very few ways to vary the flow of a game.

Sure, it often takes considerable intelligence to aim and time your shots and it can be lots of fun. But they are still mentally passive activities.

Traditional board games like backgammon or chess engage our brains more with logical thinking and strategy. They encourage creative, active thinking. Even so, things are still pretty cut and dried. How can the adventures of a checker fully capture your imagination?

A good adventure game in which you act out intrigue is far more engaging. You determine the hero's adventures until he either triumphs or is killed. You interact with a story and write it as you

go. You participate fully in the creation of the fantasy and, once hooked, might stay with the game for hours. (One software manufacturer calls its series of adventure programs "interactive fiction.")

Adventure games are ideally suited for computers. Noncomputerized adventure games have been around for quite awhile but tend to be rather awkward and complicated to play. Thus they appeal to a relatively small group of enthusiasts who are willing to tackle instruction books as long as novels. Many of these games require an extra inactive player (often called the Loremaster), who sets up the situation and plants the various monsters and treasures to surprise the main player (or players).

Note that most adventure games, computerized or not, are designed for a single hero/player. Too many creative participants can spoil the fantasy.

In computerized games, you still have to learn the basic rules, but you can simply feel your way through the details by trial, error, and logic. The computer simply won't accept an invalid move and will often prompt you for the correct move. So you do not need an extensive rule book. Figuring out the rules and what you can do with various moves can be part of the fun.

The computer also keeps track of all the necessary details—your score, the number of lurking monsters, your character's current health, etc. Computerized adventure games also eliminate complex score sheets. Of the many fine adventure game programs commercially available, the best can be a lot of fun but most suffer from significant disadvantages. One disadvantage that they all suffer from is fixed details. That is, the game is always the same with no random elements. The sack of gold is always hidden behind the bird-god statue. Solving the puzzle can be fun, getting a little closer to the final goal each time you play. But once you know the solution, there is little point in playing anymore, and the program becomes a dust collector.

Many commercially marketed games are in machine language, which speeds up the workings of the program but does not let you customize the game in any way. Many of these programs are software protected (to prevent copying), and you have no access to the program code at all.

This book hopes to provide an alternative. Good adventure games are complex and, therefore, rather tricky to program. But with some fundamental rules and a few tricks that are outlined in

the following chapters, you should be able to program your own fantasies. They are fun to write and fun to play, especially if you throw in plenty of randomness. And it's fun to swap programs with imaginative friends.

Moreover, there is a thriving market for good prewritten adventure games, so you can sell your game for a little extra cash after you've had your fun.

GETTING STARTED

All you need to get started in writing adventure game programs is a computer, your imagination and, perhaps, a little deviousness.

A printer is highly desirable so that you can get hard copies of your incomplete programs as you go along. It is far easier to spot errors in a hard copy than on a TV or video display screen. Without a printer, the job will be more tedious and time-consuming.

You will also need plenty of memory space. Rarely is 4K or 8K enough for adventure game programs unless they are quite simple and written in machine or assembly language. In this book, you work exclusively with BASIC. A little bit of inefficiency and slower calculations are acceptable. Of course, if you are an advanced programmer, you could convert the techniques discussed into assembly language.

For working in BASIC, 16K is really the bare minimum. Of the games described in the following chapters, *TREASURE HUNT* and *THE GOLDEN FLUTE* could fit into 16K; *THE GREAT ESCAPE* requires about 32K (for the complete version); and the full version of *MARS* takes close to 48K.

Of course, you can strip any of these games down to a smaller memory size if you eliminate certain features. Each game includes suggestions for reducing memory requirements.

All of the programs and examples in this book were written on a Commodore 64 computer. By comparing these commands with your computer's owner's manual, you should be able to translate them to your machine. Unusual, esoteric commands are avoided. When you write your own games, feel free to use all of the features and programming tricks in your computer's repertoire.

You don't need to be a whiz, but you should have some experience on BASIC fundamentals to benefit from this book.

You could simply type in the complete programs included here. In fact, you are invited to do so. Right there you've got a bargain, as most single adventure game programs cost more than this book.

However, you will cheat yourself if you stop with the four prewritten games. Creating adventure games is three fourths of the fun.

All techniques and tricks in this book have been tried, and are offered as suggestions. There are no absolutely right ways to program. Experiment freely. The worst that could possibly happen is you'll be left staring at an error message. Simply locate and correct the error, and try again. Every programmer has to debug. Turn your imagination loose and enjoy the fun.

Chapter 2

Creating a Plot

The first step in writing an adventure game is to determine the kind of adventure you'd like to have. The possibilities are infinite when you use your own interests and imagination.

Explore a haunted house. Rescue a medieval princess from a dungeon. Recreate an historic battle and change the course of history. Build a galactic empire. Do anything you like.

This chapter works up a sample game plot. The actual program is written in the next few chapters.

Since space-oriented games tend to be quite popular, let's write a game about exploring the planet Mars. The name of the game is MARS.

When real-life scientists explore Mars, they're sure to find plenty of scientifically interesting information. But in the adventure game sense, rock collecting doesn't thrill the average person.

Fortunately, you are not limited to the realistic or probable. So imagine there was once a glorious ancient civilization on Mars. It has long since died out, but explore the planet for valuable relics and artifacts in the course of the game.

Already you have a basis for scoring in the game. Each relic we find is worth X number of points. The object of the game is to gather up as many Martian treasures as possible, return them to the space ship, and blast off for Earth.

To create even more interest, seed the planet with worthless items that don't add to the score. In fact, to pick up a piece of junk could cost you Y points to be subtracted from the score.

You can also decide which supplies to take aboard your space ship at the beginning of the game. Supplies shouldn't have any

direct influence on the score, but they can greatly influence the explorer's chances of survival.

Arbitrarily, set up an even dozen treasures, 12 pieces of junk, and 12 supply items for a total of 36 items the player might carry. As an extra touch, make it impossible for the player to carry more than 15 items at one time. This will force the player to pick and choose carefully and plan his strategy.

An object hunt isn't too terribly exciting, so let's throw in some monsters. The player will have to deal with each type of monster in a different way to ensure variety. Add some natural obstacles like rivers, mountains, and ravines. While we're at it, weird Martian storms and/or marsquakes (the equivalent of *earthquakes*) can liven things up. Our game plot is now pretty well outlined. Things are still rather vague, but the possibilities are beginning to appear. Now fill in some of the details.

At this point you are still not pinning anything down. You can still change your mind about some things. Other ideas may occur to you as we write the program itself. But start tentatively defining things. After all, you have to begin somewhere.

THE GAME FORMAT

First off, consider the basic game structure. Until you determine the format you can't do anything else.

Since MARS is a search game through unknown territory, you should establish a playing board or map in the computer's memory. A 10-by-10 grid (see Figure 2.1) is convenient and gives you 100 areas to explore. Any more could make the game too long and drawn out and take up too much memory space. Any less would make the game too easy.

	1	2	3	4	5	6	7	8	9	10
A	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	X	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.
E	.	.	.	.	.	.	.	.	.	.
F	.	.	.	.	.	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	.
I	.	.	.	.	.	.	.	.	.	.
J	.	.	.	.	.	.	.	.	.	.

X = player's position

Figure 2.1 Playing Grid for MARS.

You can define four possible moves within the grid. North would be up (from c-5 to b-5), south would be down (from c-5 to d-5), east would be to the right (from c-5 to c-6), and west would be to the left (from c-5 to c-4).

Diagonal moves, like southwest (c-5 to d-4) could also be used but would call for more complicated programming. Since there are a lot of other things we want to put into this program, it is a good idea to conserve memory space by limiting ourselves to the four basic directional moves. If you find you have enough memory space once the program is finished to add diagonal moves, add them at that point. For the time being, however, ignore that particular possibility.

Whenever you use a playing board map format, make provision for the chance of the player moving outside the defined grid area. For example, if the player is at e-10 and moves east, his location becomes undefined. This could bomb out the program completely, or it may result in weird, unpredictable scores and plays. You must include some form of protection in any game program that uses the map format.

There are four ways to deal with this problem, and you may dream up yet another approach. Feel free to try out any ideas you come up with. The following ideas will give you a start:

1. The simplest way to deal with an off the board move would be to have the computer recognize and refuse such a move. An error message could be then displayed, such as **INVALID MOVE** or **YOU CAN'T GO THAT WAY**. The computer would then prompt the player for another move.

2. A more drastic way of dealing with an off-the-map move is for the bad move to result in instant death, or loss of the game. For example, the computer could print out, **YOU HAVE WANDERED OUTSIDE OF THE KNOWN TERRITORY. YOU WANDER AIMLESSLY ABOUT, HOPELESSLY LOST, UNTIL YOU DIE. GAME OVER.**

3. A less drastic penalty would be to randomly relocate the player somewhere within the map grid—possibly up to his neck in trouble. Similarly, in an object-gathering game like **MARS**, you could lose treasures by making an invalid move. You could then hide the forfeited treasures again within the grid.

4. The fourth approach is to simply loop around the map. For example, moving north from a-7 would position the player at j-7. Moving west from h-1 leaves the player at h-10.

Since in the game of MARS the player explores an entire planetary globe, the loop around method would be the most appropriate. This is the method used in the program.

NAMING OBJECTS

You could simply wander about, picking up object #1, and object #17, etc. Because much of the charm of adventure games comes from the imaginative details, name all objects within the game environment, being as creative as you can.

An occasional bar of gold, or diamond is OK, but it's more fun to throw in some novelty. In the case of MARS, when the player chooses between treasures and junk, some of the choices should be a little tricky and not too obviously valuable.

Table 2.1 suggests names for treasures and junk. Notice that each category has 12 items. These are the object names that will be used in the program throughout this book. If you like, you may change any or all of these.

Table 2.1 Suggested Names for Treasures and Junk Objects in the Game of MARS.

#	Treasure	Junk
1	Copper Bowl	Old Shoe
2	Gold Coins	Gaudily Ornate Ring
3	Fossilized Slide Rule	Rock
4	Statue of a Martian God	Fossilized Undershorts
5	Silver Cup	Clot of Dirt
6	Glass Orb	Old Bone
7	Scroll	Sharpened Stick
8	Glimmering Stones	Chipped Urn
9	Humming Box	Petrified Wad of Bubble Gum
10	Large Sword	Colorful Flower
11	Bleached Skull	Dead Butterfly
12	Blueprint for an Ancient Martian Palace	Indescribable Slimy Thing (?)

Creating names for the supplies is a bit more straightforward. An explorer needs certain items like food and a laser gun. However, we can also throw in a few oddball items like the old magazines. Include a few ringers—supplies that serve no purpose in the game, except to load the unwary explorer down. Remember, he can only carry up to 15 objects at a time.

Table 2.2 gives a sample list of supplies. Decide what each of these items is good for, if anything, as you write the program.

Table 2.2 Suggested Supply Objects for MARS.

#	Item
1	Food
2	Bottle of Water
3	Knife
4	Gun
5	Laser
6	Coil of Rope
7	Inflatable Raft
8	Flashlight
9	Metal Pipe
10	Old Magazines
11	Compass
12	Spacesuit

NAMING THE MONSTERS

Dreaming up monsters to plague the player can be one of the most fun parts of creating an adventure game. Let your imagination run wild.

For now, simply name the foes. You can work out their individual characteristics in a later stage of the program development.

Keep in mind that your monsters should be a varied lot. Too much of the same kind of battle, even if the opponents have different clever names, can very quickly become tedious.

Some monsters could be more powerful than others. Vary the ways they can be killed. Make a few beasts immune to, or even strengthened by, laser gun blasts. Some of the monsters could be beneficial if the player figures out how to take advantage of the situation and/or is lucky enough.

Figure 2.2 lists the monster names used in writing the sample game of MARS.

Brinchley Beast	Kufu
Ghost of an Ancient Martian	Squeanly Serpent
Grimph	Purofolee

Figure 2.2 Suggested Monsters for MARS.

NATURAL OBSTACLES

While the heart of an adventure game is usually battling assorted monsters and villains, inanimate obstacles can add to the fun and the challenge as in *MARS*, where the player explores an alien planet. Problematic landscapes are a natural feature of this game.

Set up a few mountains and rivers. Yes, even though Mars is an arid world, this is a fantasy game, so fudge a little. Throw in a few hidden ravines for the player to fall into. The fall could weaken the character and/or cause him to drop some of the objects he is carrying. You can rehide these dropped objects somewhere in the general vicinity.

Storms and marsquakes will complete the game.

SUMMARY

To create an adventure game, start by writing the story of the game. Where will it take place? Who is the hero/player? What is his goal? Are there any other characters?

In the initial stages of creating a plot, start thinking about some of the problems and obstacles that will make the player's task harder and, therefore, more interesting and fun.

For the sample game, *MARS*, the following features were considered:

- Game location—the planet Mars
- Hero/player character—an explorer from Earth
- His mission—to locate various ancient Martian treasures, and bring them aboard his spaceship to return to Earth
- Additional characters—various monsters

Now that we have a fair idea of where the game will be going, we can start writing the actual program.

Of course, you are always free to change anything in your plot as we work up the program. Sometimes you will find it difficult to implement a specific idea. Sometimes you'll come up with better ideas in midprogram. Stay flexible, although in this book the plot holds as outlined.

Even though you may change your plot at a later stage, a preliminary plot is essential. Without at least a rough blueprint, you're likely to end up with a rambling, incoherent, and pointless game that's not much fun.

Chapter 3

Beginning the Program

This chapter begins to write the program for the game MARS. Monsters and the complications will come in the next chapter. (See Figure 3.1 for programming flow chart.)

The best approach to writing a complex adventure game program is to break everything down into steps, or programming modules; then concentrate on one module at a time. This minimizes the chance of getting lost in a maze of program statements and makes the task seem less intimidating.

INITIALIZATION

First identify the program and the programmer. Do this with a simple REM (remark) statement, like this:

```
1 REM*MARS*DELTON T. HORN * V1.0
```

The first segment of a game program usually initializes the program variables. It's a good idea to identify program modules with REM statements.

Figure 3.2 shows a flow chart for the initialization module of the MARS program.

Next, dimension all of the arrays used in the program.

How many arrays do you need? You may not be sure at this stage, but that really isn't too much of a problem. You can always add more DIM statements as you realize the need for them.

Similarly, you may occasionally find that you've dimensioned an array that you end up not using. In this case, simply erase the unneeded DIM statement and recover the memory space reserved for that array.

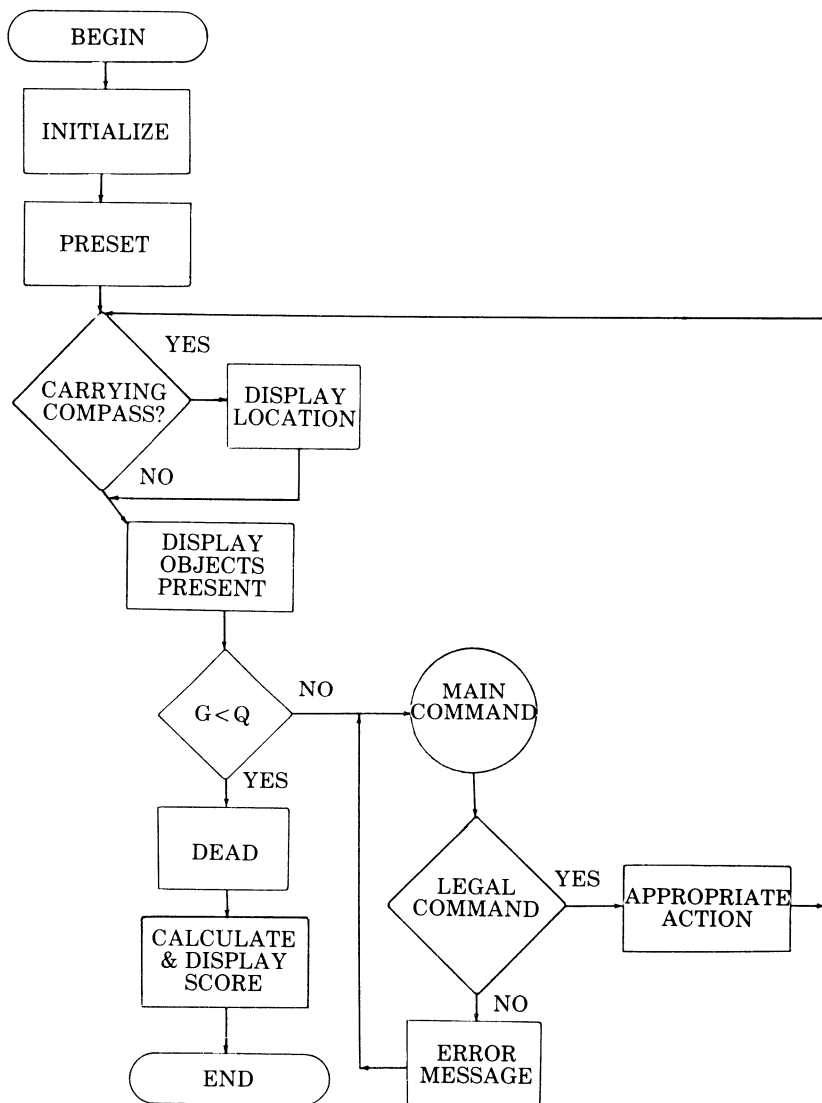


Figure 3.1 Flow-chart of the Chapter 3 Programming.

In the game of MARS, I have already worked out the necessary arrays, summarized in Table 3.1.

The initialization segment of the MARS program is Listing 3.1.

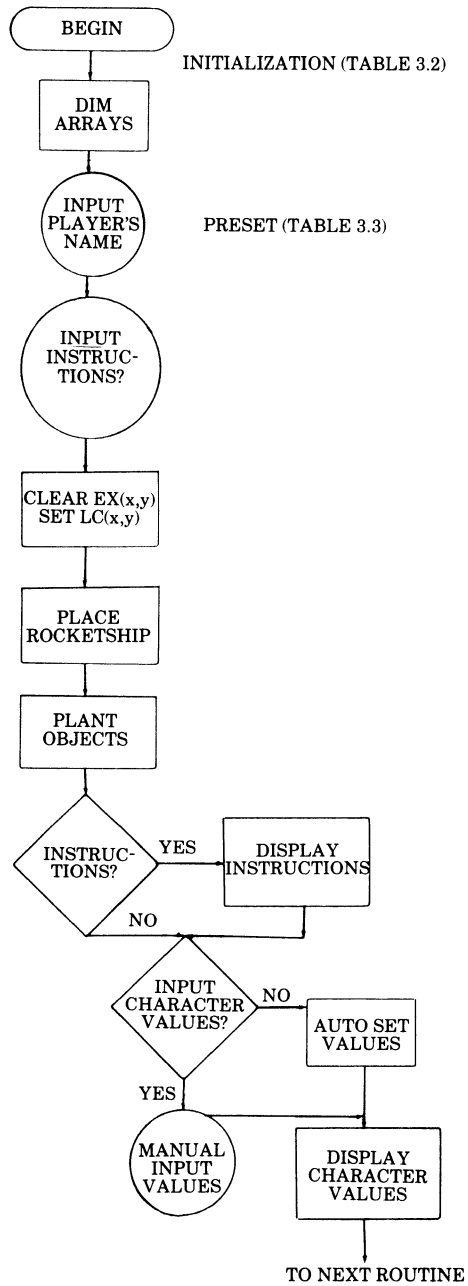


Figure 3.2 Flow-chart for Initialization and Variable Preset Routines.

Table 3.1 Arrays Used in the "Mars" Program.

array	size	function
A	12	supply horizontal location
B	12	supply vertical location
C	12	junk horizontal location
D	12	junk vertical location
E	12	treasure horizontal location
F	12	treasure vertical location
J	12	junk carried
S	12	supplies carried
T	12	treasures carried
LC	10,10	complete map
EX	10,10	explored areas map

Listing 3.1 Initialization Portion of the MARS Program.

```

1 REM*MARS*DELTON T. HORN * V1.0
5 DIM A(12), B(12), C(12), D(12), E(12),
  F(12)
10 DIM S(12), J(12), T(12), LC(10,10),
  EX(10,10)

```

INTRODUCTION AND SETTING VALUES

It is convenient, though not essential, for the computer to display some kind of introduction to the program. In the case of a fairly complex game program like MARS, it will take some time for all of the variables to be preset. This makes an introductory display even more desirable. If the screen simply remains blank until all of the variables are preset, the player would wonder if the program is running, or if the computer has locked up in error condition. The player could not tell without BREAKing the program, or just waiting. An introductory display, printed out correctly, however, would indicate that the program is indeed running properly.

The variable preset routine for the game of MARS is shown in Listing 3.2. As the flowchart in Figure 3.2 demonstrates, this section of the program is fairly straightforward.

Listing 3.2 Variable Preset Routine for the MARS Program

```
20 PRINTCL$, "MARS":PRINT:INPUT"YOUR  
   NAME";N$  
30 PRINT:INPUT "WILL YOU NEED  
   INSTRUCTIONS";Q$:Q$ = LEFT$(Q$,1)  
35 IF Q$ = "Y" THEN GOSUB 10010  
36 PRINTCL$  
40 FOR X=1 TO 10: FOR Y=1 TO 10:  
   EX(X,Y)=0  
50 PRINT "*";:Z=INT(RND (1)*17+1): IF  
   Z>12 THEN Z=0  
60 LC(X,Y)=Z: NEXT: NEXT  
70 PRINT "PLEASE BE PATIENT,";N$  
80 R1=INT(RND (1)*10+1):R2=INT(RND (1)*  
   10+1): L1=R1:L2=R2  
90 EX(R1,R2)=20:LC(R1,R2)=20: PRINT:  
   PRINT "I'M BUILDING AN ENTIRE PLANET  
   HERE!"  
100 PRINT:FOR X=1 TO 12:A(X)=R1:B(X)=R2  
110 Y=INT(RND (1)*10+1):Z=INT(RND (1)*  
   10+1):IF Y=R1 AND Z=R2 THEN 110  
120 C(X)=Y:D(X)=Z  
130 Y=INT(RND (1)*10+1):Z=INT(RND (1)*  
   10+1):IF Y=R1 AND Z=R2 THEN 130  
140 E(X)=Y:F(X)=Z  
150 S(X)=0:J(X)=0:T(X)=0  
160 NEXT:GOSUB 10000:INPUT "PLEASE PRESS  
   'RETURN' ";Q$  
170 PRINTCL$  
175 PRINT  
180 PRINT "ENTER 1 FOR AUTOMATIC  
   CHARACTER OR 2 TO CREATE YOUR  
   OWN":INPUT X  
190 IF X = 1 THEN 210  
200 IF X = 2 THEN 230  
205 GOTO 180  
210 AX = INT (RND (1) * 50 + 1) + 50  
   :DX = INT (RND (1) * 100 + 1) +  
   100
```

```

215 SX = INT (RND (1) * 50 + 1) + 50
    PX = INT (RND (1) * 50 + 1) + 50
220 GOTO 300
230 INPUT "HEALTH? ";DX: IF DX < 1 OR
    DX > 100 THEN 230
240 DX = DX * 2: INPUT "SPEED";SX: IF
    SX < 1 OR SX > 100 THEN 240
250 INPUT "POWER ";PX: IF PX < 1 OR
    PX > 100 THEN 250
260 INPUT "AIM ";AX: IF AX < 1 OR AX >
    100 THEN 260
300 PRINTCHR$(147): PRINT : PRINT
    " ",N$: PRINT
310 PRINT "HEALTH",DX / 2;"%"
320 PRINT "SPEED",SX;"%"
330 PRINT "POWER",PX;"%"
340 PRINT "AIM",AX;"%"
370 DG = DX
380 INPUT "PLEASE PRESS 'RETURN' TO PLAY
    ";Q$: PRINT CLS$ : PRINT : PRINT
9999 STOP
10000 FOR TT=1 TO 321:NEXT:RETURN
10010 PRINT "INSTRUCTIONS NOT READY":
    RETURN

```

First clear the screen, print out the name of the game, and ask for the player's name. Extra blank lines printed out make the display neater and easier to read. The player is also asked if he will want instructions for the game. This is all done in lines 20 through 36:

```

20 PRINTCL$, "MARS":PRINT:INPUT"YOUR NAME";N$
30 PRINT:INPUT "WILL YOU NEED INSTRUCTIONS?
    ";Q$:Q$=LEFT$(Q$,1)
35 IF Q$ = "Y" THEN GOSUB 10010
36 PRINT CL$

```

The player's name is assigned to the variable N\$, which will be used throughout the game.

The string variable Q\$ takes on different values throughout the program. It is used to record most of the player's commands. By

reusing the same variable for a number of temporary values, you can save a considerable amount of memory space.

Q\$ is reduced to just its first letter with the command Q\$ = LEFT\$(Q\$,1). This variable will now contain a value of Y if the player responds YES (or YEAH, or YEP, etc.). The computer will check and act upon the contents of this variable in a few lines.

On line 35, we check Q\$ and display or skip the instructions as appropriate. Notice that this subroutine is placed immediately after the time delay subroutine already entered.

The next step is to clear the explored map and plant the various random obstacle marker values throughout the main location map. The explored map is represented by the array EX(10,10), and the main location map is stored in array LC(10,10). Since these are two-dimensional arrays of equal size, combine the two operations in a single pair of nested loops (X and Y). For each step through the loops, set the value of EX(X,Y) to zero. A random number (Z) is selected. This number may have a value of 1 to 17, but if the value is greater than 12, it is set back to 0 to represent a clear (no obstacle) space in the map. A value of 0 is five times more likely than any other specific value, but the odds are 12 to 5 that any given map location will contain some obstacle. A series of asterisks is printed to reassure the player that the program has not gotten latched up. You may omit the PRINT statement in line 50, but do not omit the rest of the line.

All of this is programmed in three lines, numbered 40 through 60:

```
40 FOR X=1 TO 10: FOR Y=1 TO 10:
    EX(X,Y)=0
50 PRINT "*"; Z=INT(RND (1)*17+1):
    IF Z>12 THEN Z=0
60 LC(X,Y)=Z: NEXT: NEXT
```

The explorer's rocket ship serves as home base for this game. Remember, the object is to return the treasures to the ship and blast off to Earth. Determine a location for the rocket ship. It could always be at a fixed point, such as location 1,1, but it's more interesting to have a different randomly selected landing point for each game you play.

Since you are dealing with a two-dimension map grid, identify the rocket ship's location with two simple variables, called R1 and

R2. Similarly, the player's current location will be stored as L1 and L2. Since the explorer naturally starts out aboard his rocket ship, begin the game with L1 = R1 and L2 = R2.

The rocket ship's location should be marked in the map arrays. After all, we don't want to have a grimph attack on board the ship. We can do this by inserting the dummy obstacle value 20 into the appropriate map location. Any earlier value will be replaced by this value:

```

70 PRINT "PLEASE BE PATIENT, ";N$
80 R1 = INT(RND (1)*10+1):R2 =
    INT(RND (1)*10+1): L1=R1:L2=R2
90 EX(R1,R2) = 20:LC(R1,R2) = 20:
    PRINT:PRINT "I'M BUILDING AN
    ENTIRE PLANET HERE!":PRINT

```

Moving on down the flow chart, the next step is to plant the supply, junk, and treasure items. Since there are 12 of each stored in arrays, use a FOR...NEXT... loop to step through each one.

The supply object locations are stored in arrays A(x) and B(x). Since the supplies should naturally start out aboard the rocket ship, simply insert the values of R1 and R2 into each space in these arrays:

```

100 PRINT:FOR X=1 TO 12:A(X)=R1:B(X)=R2

```

The junk item locations are stored in arrays C(x) and D(x). Scatter these objects randomly throughout the map area, but not aboard the ship. An IF...THEN... check causes the computer to select new map coordinates if it happens to duplicate the rocket ship's location:

```

110 Y = INT(RND (1)*10+1):Z = INT(RND
    (1)*10+1):IF Y=R1 AND Z=R2
    THEN 110
120 C(X)=Y:D(X)=Z

```

Plant the treasure objects in the same way, except use arrays E(x) and F(x):

```

130 Y = INT(RND (1)*10+1):Z = INT(RND
    (1)*10+1):IF Y = R1 AND Z = R2
    THEN 130
140 E(X)=Y:F(X)=Z

```

Notice that there is no provision to prevent multiple junk and/or treasure items from appearing in a single location. Statistically, this won't happen very often.

Three additional 12-space arrays (S(x), J(x), and T(x)) are used to keep track of the items the explorer character is carrying. Since at the start of the game he shouldn't be carrying anything, we'll place zeros into each of these array locations, and close the loop with the NEXT statement:

```
150 S(X)=0:J(X)=0:T(X)=0
160 NEXT:GOSUB 10000:INPUT "PLEASE PRESS
    'RETURN' ";Q$
```

Line 160 also calls a subroutine. This is a simple timing delay loop:

```
10000 FOR TT=1 TO 321:NEXT:RETURN
```

The time delay subroutine included here gives the player a better chance to appreciate the humorous message in lines 70 and 90. Eliminate the GOSUB command from this line if you prefer, but be sure to type in the subroutine line itself. Use the time delay subroutine throughout the program.

The structure of a computer program is usually clearer if the subroutines are kept separate from the main program. Usually, the subroutines are placed after the rest of the program. By placing the first subroutine at line 10000, you can be reasonably sure you'll have enough lines open for the main body of the program.

The line number 10000 is somewhat arbitrarily chosen, but is a good choice because it is easy to remember. Since in most game programs, a time delay subroutine will probably be the most frequently called, place it first at the easy-to-remember address.

You should add a STOP statement just before the subroutines begin to prevent the program from accidentally crashing through to a RETURN without a GOSUB command. This can occur through an error in the programming or in sample test runs of the incomplete program. Ideally you should be able to remove this line from the finished program, but it usually stays. By giving the STOP command an odd number (9999 is used here), the beginning of the subroutines is easier to find in the line listing.

Returning to line 160, there is also an INPUT command that requests the player to press 'RETURN' without entering any data.

This handy little trick lets the player manually determine how long the current information will be displayed on the screen.

You need a string variable to accept the null input. On the Commodore 64, a null input (pressing RETURN without typing in anything) leaves the variable with its previous value without any change. The Q\$ should still hold the answer to the instructions question of line 30. If you feel uncomfortable with this, use another variable name in line 160.

Since you often don't know all the details of a game this early in the programming, it is a good idea to leave actual instruction writing until later. The instructions for MARS are handled in a later chapter.

For now, just include a dummy subroutine to prevent bombing out during program test runs:

```
10010 PRINT "INSTRUCTIONS NOT READY:"  
      RETURN
```

To make the game more interesting, the character's success at various tasks can depend on certain characteristics changed from game to game. For MARS, assign ratings up to 100 for Speed, Aim, and Power (or Strength), with the variables SX and PX.

Episodes throughout the game will affect the character's health, either decreasing it (i.e., being attacked by a monster), or increasing it (i.e., eating food). You need two health variables, one to indicate the character's maximum health rating (up to 200—also his initial rating), (DX), and his constantly changing health rating (DG).

The computer can either select random values for each of the ratings or the player can set up his own character. A beginning player can make things a little easier by entering 100, the maximum acceptable input, for each of the characteristics (DX is multiplied by two). Programming for both of these approaches is done in lines 180 through 260 in Listing 3.2.

Display the four characteristics in percent in lines 300 to 340.

The current health rating (DG) is set equal to the maximum health rating (DX) in line 370. Line 380 is another dummy input (press 'RETURN') command so the player may study the character ratings as long as he likes before beginning the game itself and clearing the display screen.

Begin keeping a chart of all subroutines at this point. So far we only have two:

```
10000 TIME DELAY
10010 INSTRUCTIONS (temporary dummy)
```

But the number of subroutines will quickly increase and become difficult to keep track of unless you take notes.

Make up a chart of all variables used in the program to prevent reusing a variable that shouldn't be changed. The variables used thus far in the MARS program are summarized in Table 3.2. Figure 3.2 is a flow chart for the initialization and variable preset portions of the program (Listings 3.1 and 3.2).

Table 3.2 Variables Used in the Initialization and Preset Routines of the MARS Program (see Listings 3.1 and 3.2).

AX	Character's Aim Rating
DG	Character's Current Health Rating
DX	Character's Maximum Health Rating
L1,L2	Current Location Coordinates
PX	Character's Power Rating
R1,R2	Rocket Ship Location Coordinates
SX	Character's Speed Rating
TT	Timing Loop
X,Y,Z	Misc. Calculations
N\$	Player's Name
Q\$	Instructions? / Dummy

Arrays used are outlined in Table 3.1.

BEGINNING THE MAIN PLAY ROUTINE

Now that the basic variables of the game have been preset, start programming the active part of the game—the play itself.

The main play routine is summarized in the flow chart of Figure 3.3 and in Listing 3.3.

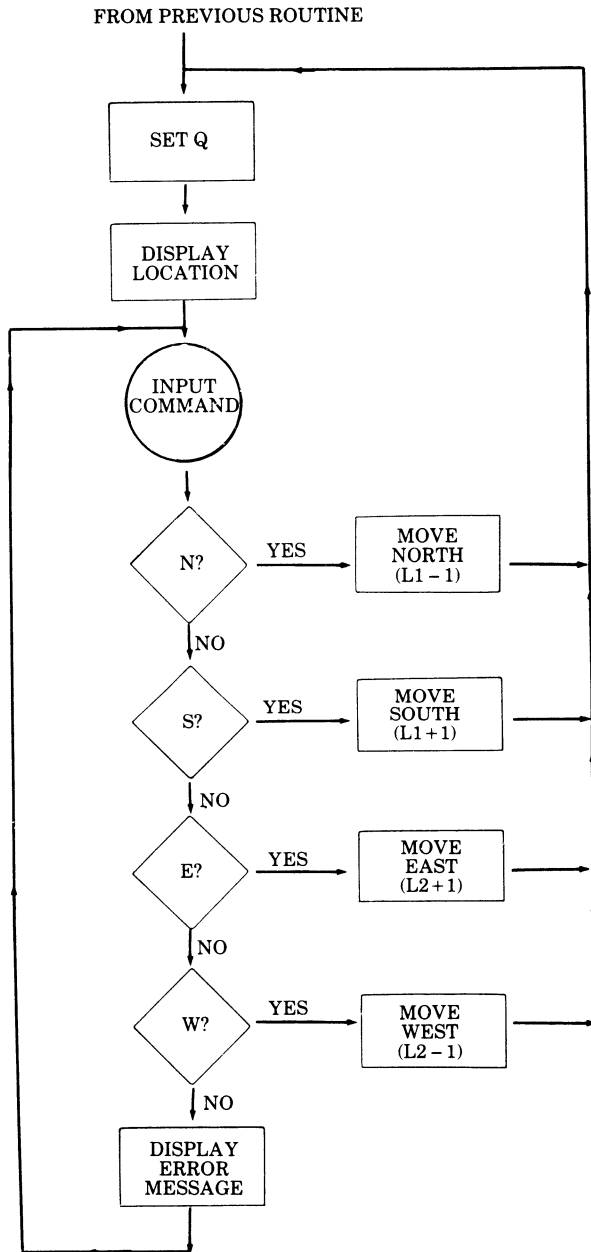


Figure 3.3 Flow-chart for Main Play Routine.

Listing 3.3 Main Play/Move/Error Message Routine for the MARS Program.

```
399 REM * LOCATION DISPLAY *
400 Q=LC(L1,L2): PRINT:PRINT "YOUR
    CURRENT COORDINATES ARE";
415 PRINT L1;";";L2
699 REM * MAIN COMMAND *
700 Q$="":DG=DG - 1: PRINT:PRINT
    "YOUR COMMAND, ";N$
710 INPUT Q$
720 IF Q$= "N" THEN 5100
730 IF Q$= "S" THEN 5120
740 IF Q$= "E" THEN 5140
750 IF Q$= "W" THEN 5160
970 GOSUB 11700
980 DG=DG-1:GOTO 500
5099 REM * MOVES * NORTH *
5100 K = 0:L3 = L1:L4 = L2:L1 =
    L1-1:IF L1<1 THEN L1=10
5110 GOTO 400
5119 REM * SOUTH *
5120 K = 0:L3 = L1:L4 = L2:L1 + 1:
    IF L1 > 10 THEN L1 = 1
5130 GOTO 400
5139 REM * EAST *
5140 K = 0:L3 = L1:L4 = L2:L2 = L2 +
    1: IF L2 > 10 THEN L2 = 1
5150 GOTO 400
5159 REM * WEST *
5160 K = 0:L3 = L1:L4 = L2:L2 = L2 -
    1: IF L2 < 1 THEN L2 = 10
5170 GOTO 400
11699 REM * ERROR *
11700 X=INT(RND (1)*8+1):IF X=1 THEN
    PRINT "SAY WHAT?"
11710 IF X=2 THEN PRINT "THAT DOES NOT
    COMPUTE."
11720 IF X=3 THEN PRINT "DON'T BE
    SILLY, ";N$;"!"
11730 IF X=4 THEN PRINT "?????"
```

```

11740 IF X=5 THEN PRINT "I DON'T
      UNDERSTAND."
11750 IF X=6 THEN PRINT "HOW WOULD
      THAT HELP HERE?"
11760 IF X=7 THEN PRINT "WHAT ON MARS
      ARE YOU BABBLING ABOUT, ";N$;"?"
11770 IF X=8 THEN PRINT "THAT DOESN'T
      MAKE ANY SENSE, ";N$;"!"
11780 RETURN

```

As the flow chart shows, each round begins by revealing the current location, that is, telling the player where on Mars he is. This can be done quite simply as follows:

```

400 Q=LC(L1,L2): PRINT:PRINT
    "YOUR CURRENT COORDINATES ARE";
415 PRINT L1;" ";L2

```

Line 400 also sets variable Q equal to the obstacle number of the main map array location.

Of course, this could be combined into a single line:

```

400 Q=LC(L1,L2): PRINT:PRINT
    "YOUR CURRENT COORDINATES ARE
    ";L1;" ";L2

```

However, you may recall from Chapter 2 that one of the supply objects is a compass. In a more advanced stage of the program, alter this routine slightly so that the coordinates are displayed only when the character carries the compass.

Once the location coordinates are displayed, ask the player for his choice of action. Remember that later on you will add a number of monsters and obstacles that have a direct effect on the player's decisions. So leave plenty of space to add this necessary programming later. Jump down to line 700 to ask the player for his move:

```

700 Q$="":DG=DG - 1: PRINT:PRINT
    "YOUR COMMAND, ";N$;
710 INPUT Q$

```

The string variable Q\$ stores the player's move. It is set to a null string (Q\$ = " ") before the command request to erase any previous commands in case the player accidentally hits the 'RETURN' key before typing in a move.

We also subtract 1 from the player's current health rating (DG). This limits how long the explorer can roam aimlessly about.

For the time being, set up the program so that it will only accept directional commands. That is, the player can only move north, south, east, or west through the map area. For simplicity, diagonal moves; i.e., southeast, are not allowed.

These commands will be in the convenient form of the initial letter only. To move north, for example, simply enter N. The directional commands can be recognized with a series of IF...THEN... statements:

```
720 IF Q$= "N" THEN 5100
730 IF Q$= "S" THEN 5120
740 IF Q$= "E" THEN 5140
750 IF Q$= "W" THEN 5160
```

The line numbers for the first command (line 5100) are arbitrarily selected. Again, this is a relatively easy to remember number and leaves plenty of space for other programming. The fact that two lines will be needed for each direction determines the other three line numbers.

Since our map area is a 10 X 10 grid, move to a new location by adding or subtracting 1 to one of the location coordinates (L1 and L2). Use L1 as the north/south counter (north is L1 - 1, and south is L1 + 1). An L2 will serve as our east/west counter (east is L2 + 1, and west is L2 - 1).

Include protection to prevent the move from going out of the boundaries defined by the map, an undimensioned array location (see Chapter 2). That is, neither counter (L1 or L2) may be less than 1, nor greater than 10. Use the loop-around method, as if the explorer goes completely around the planet. For example, if L1 becomes 11 after a move south, it changes to 1.

For some of the advanced plays later in the game, keep track of the player's previous location. Do this by setting up another pair of variables; i.e., L3 and L4. Set L3 equal to L1 and L4 equal to L2

before each new move is computed. Put together, the routine for moving north should look like this:

```
5100 K = 0:L3 = L1:L4 = L2:L1 =
      L1-1:IF L1<1 THEN L1=10
5110 GOTO 400
```

Notice that, after the move, the program loops back around to the coordinate display (line 400) to begin a new round.

Programming for moving south, east, and west works in the same way, and is given in Listing 3.3.

The command request routine so far will recognize inputs of N, S, E, or W. What happens if the player enters something else? The program will need to display an error message if this happens and then loop back around to request a new command.

For variety, use eight different error messages, any one of which may be randomly selected. Put this into a subroutine to get error messages at other points in the game, too.

Before defaulting to the error subroutine, leave space for additional commands as you add complexity to the game. Therefore, you might number these steps like this:

```
970 GOSUB 11700
980 DG=DG-1:GOTO 500
11700 X=INT(RND (1)*8+1):IF X=1
      THEN PRINT [error message #1]
11710 IF X=2 THEN PRINT
      [error message #2]
      * * *
11770 IF X=8 THEN PRINT
      [error message #8]
11780 RETURN
```

Alternatively, the subroutine could be programmed as follows:

```
11700 X=INT(RND (1)*8+1):ON X GOTO
      11710, 11720, 11730, 11740, 11750,
      11760, 11770, 11780
11710 PRINT [error message #1]:RETURN
11720 PRINT [error message #2]:RETURN
      * * *
11780 PRINT [error message #8]:RETURN
```

Either of these methods produces exactly the same results. Some programmers prefer the `ON X GOTO . . .` method as more elegant and faster running. However, in this particular case, when only a single `PRINT` command is dependent on the value of `X`, the difference in operating speed would be an unnoticeably small fraction of a second. The multiple `IF . . . THEN . . .` statements of the first method are clearer and easier to type in. The long string of numbers required for the `ON X GOTO . . .` statement invites typing errors.

When several commands are dependent on the value of `X` (or any variable), the `ON X GOTO . . .` method is certainly the way to go. In this instance, the choice is really just a matter of personal preference.

Another place that allows personal preference is in the error messages themselves. Use the messages as given in Listing 3.3, or let your imagination run free as you come up with your own. Good error messages add a lot to the program's personality. Just saying `INVALID MOVE` is rather prosaic.

Remember, however, that it is annoying to play the game if the error messages are too insulting, especially if someone other than the original programmer runs the program. Use wit, not abuse.

ADDING COMMANDS

At this stage, the game is extremely dull and pointless. The player can only wander aimlessly about. So add more commands to supplement the four directional commands.

The completed game of `MARS` will use the 20 additional commands listed in Table 3.3. If you were programming your own game, you might not be sure of all of the commands you'll want to use yet, but you should make a list of the commands you expect to use. Later you can add more and/or eliminate others.

Table 3.3 Commands Used in the Game of `MARS`.

SCORE	DIAGNOSIS	CRY
EAT	DRINK	FILL
INFLATE	INVENTORY	CLIMB
LOOK	OPEN	PRAY
GET	DROP	KILL
HELP	WAIT	TOUCH
MAP	BLAST OFF	

In discussing the programming for these commands, you will notice that the line numbers jump around a bit. This is because I did not originally write the command routines in the order in which they are discussed.

First, to save a little memory space and typing, chop off the input command (Q\$) to its first three letters (QX\$) in the command identification statements. Some commands will require an object, that is, you need two words such as GET ROCK. Use the last three letters of the original command (QY\$) to identify the object of the command. Before you check for the new commands, form the required substrings:

```
760 QX$ = LEFT$ (Q$, 3) : QY$ =  
    RIGHT$ (Q$, 3)
```

The entire original command is still stored as Q\$, in case more information is needed from it.

Listing 3.4 has the programming for adding three new commands—SCORE, DIAGNOSIS, and CRY. These routines are flow-charted in Figure 3.4.

First, set up the scoring procedure. Treasure objects add to the score; junk objects subtract from it. The amount added or subtracted will be greater if the objects are dropped aboard the rocket ship. To summarize the scoring, say:

```
Carrying a treasure object = +10  
Dropping a treasure object aboard the rocket ship = +12  
Carrying a junk object = - 2  
Dropping a junk object aboard the rocket ship = - 3.5
```

Add points for monsters killed in the game. The variable SV keeps track of this. Of course SV will always be equal to 0 until we add the monsters themselves (see the next chapter), but put it into the score routine now to make sure you don't forget it.

Don't bother to keep a running score each time an object is picked up or dropped. The programming could be a little awkward and a running score isn't really necessary anyway. It's easy enough to calculate the object score and add SV whenever the score is needed.

To calculate the score, check six of the arrays: J(x) (junk carried), T(x) (treasures carried), C(x) and D(x) (junk locations), and E(x) and F(x) (treasure locations). Since all of these are 12-location arrays, you can check all within a single 12 step FOR . . . NEXT loop. The

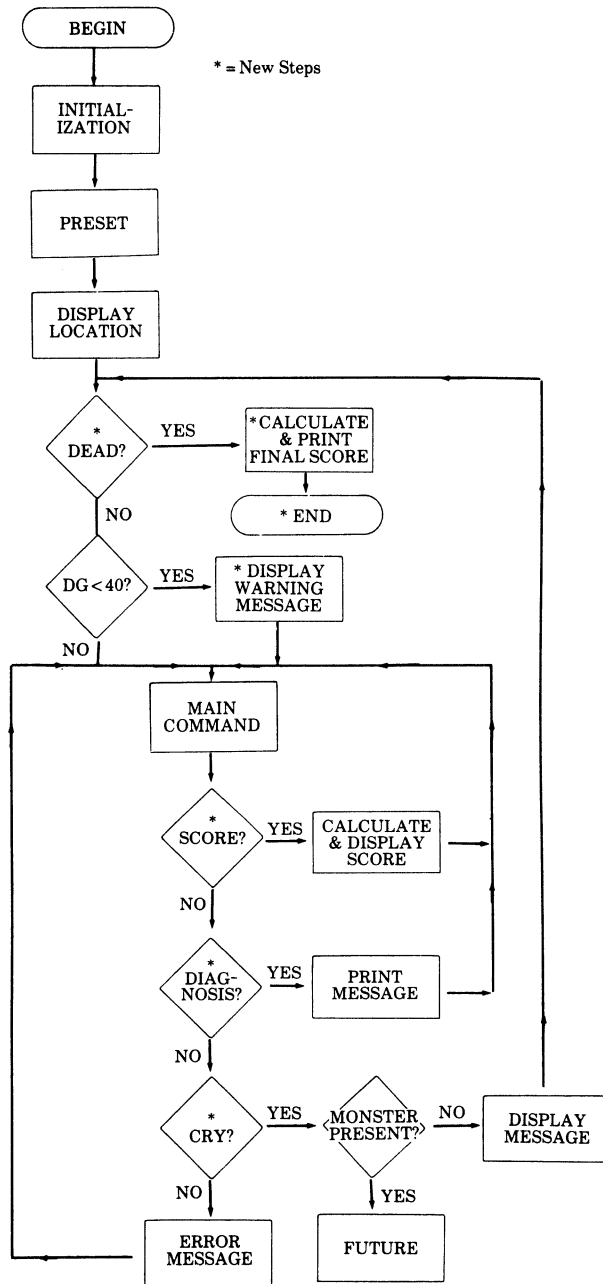


Figure 3.4 Flow-chart for SCORE/DIAGNOSIS/CRY.

calculation will be performed in a subroutine so it can easily be accessed at various points throughout the program.

```

10400 SC=0:FOR X=1 TO 12:IF
      T(X)=1 THEN SC=SC+10
10410 IF E(X)=R1 AND F(X)=R2 THEN
      SC=SC+12
10420 IF J(X)=1 THEN SC=SC-2
10430 IF C(X)=R1 AND D(X)=R2 THEN
      SC=SC-3.5
10440 NEXT :SC=SC+SV: RETURN

```

The total score is now stored as SC.

Listing 3.4 Adding the SCORE, DIAGNOSIS and CRY Commands to the MARS Program.

```

649 REM * HEALTH CHECK *
650 IF DG < 1 THEN 5070
660 IF DG < 40 THEN PRINT "YOU'RE NOT
      LOOKING TOO WELL, PAL."
760 QX$ = LEFT$(Q$,3):QY$ = RIGHT$
      (Q$,3)
770 IF QX$ = "SCO" THEN GOSUB 10400 :
      PRINT "YOUR SCORE SO FAR IS "; SC:
      GOTO 700
780 IF QX$="DIA" THEN GOSUB 11150:GOTO 700
790 IF QX$="CRY" THEN 5180
5070 GOSUB 10000:PRINT:PRINT "YOU ARE
      DECEASED.":PRINT
5080 PRINT "YOUR FINAL SCORE WAS ";
5090 GOSUB 10400:PRINT SC:END
5179 REM*CRY*
5180 REM*HOLD SPACE TO ADD MONSTERS*
5200 PRINT "WHY? ARE YOU UPSET FOR SOME
      REASON?"
5210 GOTO 500
10399 REM * SCORE CALCULATION *
10400 SC = 0: FOR X = 1 TO 12: IF T(X)
      = 1 THEN SC = SC + 10
10410 IF E(X) = R1 AND F(X) = R2 THEN
      SC = SC + 12

```

```
10420 IF J(X) = 1 THEN SC = SC - 2
10430 IF C(X) = R1 AND D(X) = R2 THEN
    SC = SC - 3.5
10440 NEXT :SC = SC + SV: RETURN
11149 REM * DIA *
11150 X = DX * .8: IF DG > X THEN PRINT
    "YOU'RE FEELING JUST FINE & DANDY!":
    RETURN
11160 IF DG > 150 THEN PRINT "YOU FEEL
    VERY GOOD.":RETURN
11170 IF DG > 120 THEN PRINT "YOU FEEL
    PRETTY GOOD.":RETURN
11180 IF DG > 105 THEN PRINT "YOU'RE NOT
    FEELING TOO BAD, ALL THINGS
    CONSIDERED."
11185 IF DG > 105 THEN RETURN
11190 IF DG > 90 THEN PRINT "SOME ALKA-
    SELTZER MIGHT BE NICE...": RETURN
11200 IF DG > 80 THEN PRINT "YOU'VE HAD
    BETTER DAYS.": RETURN
11210 IF DG > 70 THEN PRINT "YOU'RE IN
    NO SHAPE TO GO DANCING.": RETURN
11220 IF DG > 60 THEN PRINT "YOU'RE
    FEELING RATHER POORLY.": RETURN
11230 IF DG > 50 THEN PRINT "ARE YOUR
    INSURANCE PAYMENTS UP TO DATE?":
    RETURN
11240 IF DG > 40 THEN PRINT "BETTER
    REHEARSE YOUR MOANS AND GROANS.":
    RETURN
11250 IF DG > 30 THEN PRINT "YOU'RE
    NOT DOING WELL AT ALL.": RETURN
11260 PRINT "IT'S A WONDER YOU CAN
    STILL STAND UP!": RETURN
```

Now add a line to check for the SCORE command as the input. If it is found jump to the subroutine, print the result, and loop back around to ask for a new command:

```
770 IF QX$= "SCO" THEN GOSUB 10400:
    PRINT "YOUR SCORE SO FAR IS ";SC:
    GOTO 700
```

The SCORE command is now programmed.

Use the variable DG to keep track of the character's current health. A number of factors throughout the game will affect this value. Obviously, if the character's health rating drops down to 0 or lower, consider him dead. Check this before the command request:

```
650 IF DG < 1 THEN 5070
5070 GOSUB 10000:PRINT:PRINT "YOU
    ARE DECEASED.":PRINT
```

Subroutine 10000 is the timing delay already introduced.

To avoid frustration and to let the player know how he was doing up to the time of his demise, jump to the score calculation subroutine and print the result before ending the program:

```
5080 PRINT "YOUR FINAL SCORE WAS ";
5090 GOSUB 10400:PRINT SC:END
```

We should warn the player if his health rating gets too low. Line 660 prints out a warning message if the value of DG dips below 40:

```
660 IF DG < 40 THEN PRINT "YOU'RE NOT
    LOOKING TOO WELL, PAL."
```

The DIAGNOSIS command (or DIA for short) allows the player to check the health rating whenever he wants throughout the game. This allows him to plan his strategy better. We could just print out the value of DG:

```
780 IF QX$="DIA" THEN PRINT "CURRENT
    HEALTH RATING IS ";DG:GOTO 700
```

but that isn't very interesting. I prefer to use a subroutine that will print out an appropriate message depending on what range of values DG falls in:

```
780 IF QX$="DIA" THEN GOSUB 11150:
    GOTO 700
11150 X = DX * .8: IF DG > X THEN
    PRINT [message #1]:RETURN
```

```
11160 IF DG > 150 THEN PRINT
      [message #2]:RETURN
11170 IF DG > 120 THEN PRINT
      [message #3]:RETURN
      * * *
11250 IF DG > 30 THEN PRINT
      [message #11]:RETURN
11260 PRINT [message #12]:RETURN
```

The first message will be printed if the DG value is better than 80 percent of the maximum health rating (DX). The other ranges are based on absolute numbers, 150, 120, etc., but you can change these check-points if you prefer.

Placing a RETURN statement after each PRINT statement ensures that only a single message will be displayed, regardless of the value of DG.

Message 12 (line 11260) will be displayed only if DG is less than 30. An IF...THEN... test is not required here, because the earlier lines will have already taken care of any higher DG values.

Write your own messages, or use the ones I came up with, as they are shown in Listing 3.4.

A good adventure game will result in a certain degree of frustration on the part of the player. He may want to just sit down and cry if things get too rough. Why not let the computer recognize CRY as a command, even though it is not likely to help the situation much. CRYing could have special results when certain monsters are present, so we'll leave some space for the appropriate IF...THEN... tests. Since these would come at the beginning of the routine, put a dummy statement at the line called by the GOTO command:

```
790 IF QX$="CRY" THEN 5180
5180 REM*HOLD SPACE TO ADD MONSTERS*
```

Ordinarily you should never have a GOTO or GOSUB command reference a line number containing only a REM (remark) statement. If the program gets too long, the REMs are the first thing to eliminate (since they don't affect the way the program runs anyway). Going through the complete line listing and correcting GOTO or GOSUB statements is tedious at best, and you may miss one or two, causing the program to bomb.

To make sure you don't actually reference a REM line, use odd line numbers for all of your REM statements. For instance:

```
5179 REM*CRY*
```

The active commands are at line numbers that are multiples of 10 (for the most part), so we know just by the line number that this is a REM line.

However, when temporarily holding a line in an incomplete program, as we are doing here, it is OK to reference a REM line from a GOTO or GOSUB statement. The REM statement will be replaced by an active command later.

Returning to the CRY routine, if no monsters are present, do not let CRYing have any particular effect. The computer will just display a sarcastic message, then loop back around to ask for a new command:

```
5200 PRINT "WHY? ARE YOU UPSET FOR SOME  
      REASON?"  
5210 GOTO 500
```

Of course, you may wish to have the computer permit a different message.

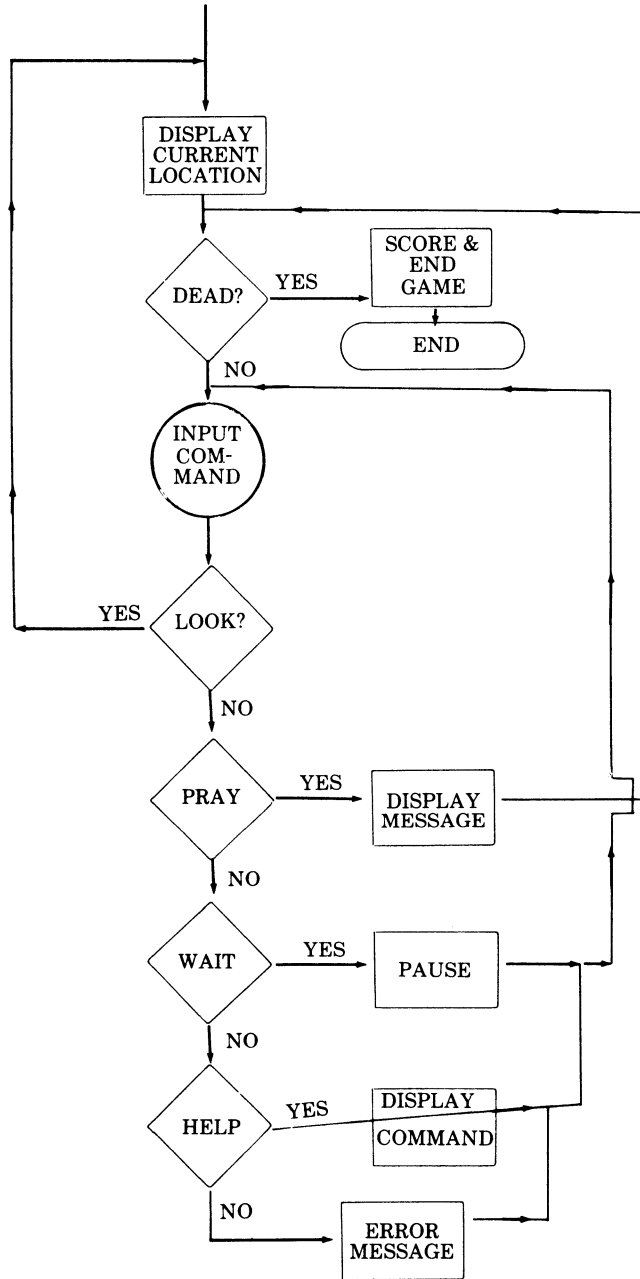
Tossing in a few inconsequential commands like CRY can increase the entertainment value of an adventure game, especially when the player doesn't know what will happen.

Listing 3.5 contains the programming for four more commands for the game of MARS. The new commands are LOOK, PRAY, HELP, and WAIT. As the flow-chart of Figure 3.5 shows, these commands are all quite straightforward.

As the game grows increasingly more complex, the player may select a number of actions before moving on to a new location. The current coordinates may well be scrolled off the top of the display screen and the player could lose his location. Permit a simple command that allows the player to reorient himself. A command of LOOK causes the program to loop around and redisplay the current coordinates:

```
860 IF QX$= "LOO" THEN 400
```

The PRAY command is similar to the CRY command introduced in the last section of this chapter. It will have special results when

**Figure 3.5 Flow-chart for LOOK/HELP/PRAY/WAIT.**

Listing 3.5 LOOK, HELP, PRAY, and WAIT Commands for the MARS Program.

```

860 IF QX$ = "LOO" THEN 400
880 IF QX$ = "PRA" THEN 6550
920 IF QX$ = "HEL" THEN GOSUB 11800 :
    GOTO 700
930 IF QX$ = "WAI" THEN 9820
6550 REM*HOLD SPACE FOR MONSTERS*
6590 DG=DG+1:PRINT:PRINT "GIMME THAT
    OLD TIME RELIGION!":PRINT:GOTO 500
9820 PRINT " ", "(PAUSE)": PRINT: GOSUB
    10000
9830 DG = DG + 10: IF DG > DX THEN DG
    = DX
9840 GOTO 500
11799 REM * HELP *
11800 PRINT : PRINT "POSSIBLE COMMANDS
    INCLUDE --- "
11810 PRINT "BLAST OFF", "CLIMB", "CRY",
    "DIA", "DRINK",
11820 PRINT "DROP", "EAT", "FILL", "GET",
    "HELP", "INFLATE",
11830 PRINT "INV", "KILL", "LOOK",
    "MAP", "OPEN",
11840 PRINT "PRAY", "SCORE", "TOUCH",
    "WAIT"
11850 PRINT : PRINT "NOT ALL COMMANDS
    WILL BE RECOGNIZED AT ALL TIMES."
11860 PRINT : INPUT "PLEASE PRESS 'RETURN'
    TO CONTINUE THE GAME ";Q$
11870 RETURN

```

facing specific monsters, but when no monster is present, only a nonsensical message is displayed. The current health rating (DG) is also increased by 1, cancelling out the normal exertion of making a command (see line 700):

```

880 IF QX$= "PRA" THEN 6550
6550 REM*HOLD SPACE TO ADD MONSTERS*
6590 DG=DG+1:PRINT:PRINT "GIMME THAT OLD
    TIME RELIGION!":PRINT:GOTO 650

```

Another simple command that might prove useful would be one to call up a list of all possible commands for use in the game. Some users may prefer to leave out this command and search for acceptable commands in a hit or miss fashion. Some players consider this part of the fun of learning a new adventure game. That is perfectly all right.

For those who choose to include it, the **HELP** command is included in Listing 3.5. A subroutine is used to print out the list of possible instructions. The subroutine as given in the table includes all of the commands that will be used in the final version of the game as presented in this book. You may want to add some new ones of your own, or delete one or two. It's easy enough to add extra **PRINT** commands to this simple subroutine.

Note that the **HELP** subroutine includes a message stating that not all of the commands will be valid under all circumstances (line 11850). For example, you can't **INFLATE** your raft if you are not carrying it.

So far we have been decrementing the health rating (**DG**) on each command (except **PRAY**), but we haven't allowed any way for the character to recover. One way to recover would be to stop and rest for a while. The **WAIT** command allows the player to choose this option.

When the **WAIT** command is entered, (pause) is displayed and the timing subroutine (10000) is called. The **DG** current health rating is increased by 10. An **IF . . . THEN** test is included in line 9830 to prevent the current health rating (**DG**) from exceeding the character's maximum health level (**DX**).

After all of this is done, the program loops back around to ask for a new command:

```
930 IF QX$= "WAI" THEN 9820
9820 PRINT " ", "(PAUSE)":PRINT:GOSUB
      10000
9830 DG=DG+10: IF DG > DX THEN DG=DX
9840 GOTO 500
```

FINDING THE OBJECTS

Since the object of the game of **MARS** is to collect ancient Martian treasures, you obviously need a routine to identify these objects when you encounter them.

Three pairs of 12 locations check against the player's current coordinates. Do this with a simple FOR...NEXT... loop, like this:

```

440 Y=0: FOR X=1 TO 12: IF A(X)=L1 AND
    B(X)=L2 THEN GOSUB 10450
450 IF C(X)=L1 AND D(X)=L2 THEN GOSUB
    10570
460 IF E(X)=L1 AND F(X)=L2 THEN GOSUB
    10700
480 NEXT X

```

The purpose of the variable Y will be explained shortly.

Each of the subroutines called from this loop will print out one of 12 messages (determined by the current value of X) to identify the object present.

Since many different objects may be at a single location, especially aboard the rocket ship in later stages of the game, some lines could be scrolled off the top of the screen before the player has a chance to read them. The variable Y is used as a counter to eliminate this problem.

Each time one of the subroutines is called, it adds 1 to Y before returning control to the main body of the program, for example:

```

10565 Y=Y+1: RETURN

```

If Y accumulates a value greater than 8, it is set back to 0 and the computer stops until the player presses the 'RETURN' key, indicating he is ready for more information:

```

470 IF Y > 8 THEN Y=0:PRINT:INPUT "PLEASE
    PRESS 'RETURN'      ";Q$:PRINT

```

You might want to try a sample run at this point. Make sure that all 12 supply items appear at the beginning location (aboard the rocket ship), and that the treasure and junk items are scattered randomly throughout the rest of the map. Try out all of the commands added since the last sample run.

Well, now you can see the various objects, but you can't do anything with them. A new command, GET, will allow the explorer to pick up the various items. Objects can be gotten rid of by using the DROP command.

Listing 3.6 Routines for Displaying Object Locations in the MARS Program.

```
440 Y=0: FOR X=1 TO 12: IF A(X)=L1 AND
    B(X)=L2 THEN GOSUB 10450
450 IF C(X)=L1 AND D(X)=L2 THEN GOSUB
    10570
460 IF E(X)=L1 AND F(X)=L2 THEN GOSUB
    10700
480 NEXT X
10449 REM * SUPPLIES PRESENT *
10450 IF X = 1 THEN PRINT "SOME FOOD
    IS HERE."
10460 IF X = 2 THEN PRINT "A BOTTLE OF
    WATER IS HERE."
10470 IF X = 3 THEN PRINT "A KNIFE IS
    HERE."
10480 IF X = 4 THEN PRINT "A GUN IS
    HERE."
10490 IF X = 5 THEN PRINT "A LASER IS
    HERE."
10500 IF X = 6 THEN PRINT "A COIL OF
    ROPE IS HERE."
10510 IF X = 7 THEN PRINT "AN INFLATABLE
    RAFT IS HERE."
10520 IF X = 8 THEN PRINT "A FLASHLIGHT
    IS HERE."
10530 IF X = 9 THEN PRINT "A METAL PIPE
    IS HERE."
10540 IF X = 10 THEN PRINT "SOME OLD
    MAGAZINES ARE HERE."
10550 IF X = 11 THEN PRINT "A COMPASS
    IS HERE."
10560 IF X = 12 THEN PRINT "YOUR
    SPACESUIT IS HANGING NEATLY ON ITS
    RACK."
10565 Y = Y + 1: RETURN
10569 REM * JUNK PRESENT *
10570 IF X = 1 THEN PRINT "AN OLD
    SHOE IS HERE."
10580 IF X = 2 THEN PRINT "A GAUDILY
    ORNATE RING IS HERE."
```

```
10590 IF X = 3 THEN PRINT "A ROCK IS  
HERE."  
10600 IF X = 4 THEN PRINT "A PAIR OF  
FOSSILIZED UNDERSHORTS IS HERE."  
10610 IF X = 5 THEN PRINT "A LARGE CLOT  
OF DIRT IS HERE."  
10620 IF X = 6 THEN PRINT "AN OLD BONE  
IS HERE."  
10630 IF X = 7 THEN PRINT "A SHARPENED  
STICK IS HERE."  
10640 IF X = 8 THEN PRINT "A BADLY  
CHIPPED URN IS HERE."  
10650 IF X = 9 THEN PRINT "A PETRIFIED  
WAD OF BUBBLE GUM IS HERE."  
10660 IF X = 10 THEN PRINT "A COLORFUL  
FLOWER IS HERE."  
10670 IF X = 11 THEN PRINT "A DEAD  
BUTTERFLY IS HERE."  
10680 IF X = 12 THEN PRINT "AN  
INDESCRIBABLE SLIMY THING IS HERE."  
10690 Y = Y + 1: RETURN  
10699 REM * TREASURES PRESENT *  
10700 IF X = 1 THEN PRINT "A DENTED  
COPPER BOWL IS HERE."  
10710 IF X = 2 THEN PRINT "SOME GOLD  
COINS ARE HERE."  
10720 IF X = 3 THEN PRINT "A FOSSILIZED  
SLIDE RULE IS HERE."  
10730 IF X = 4 THEN PRINT "A STATUE OF A  
THREE-ARMED MARTIAN GOD IS HERE."  
10740 IF X = 5 THEN PRINT "A TARNISHED  
SILVER CUP IS HERE."  
10750 IF X = 6 THEN PRINT "A GLASS ORB  
IS HERE."  
10760 IF X = 7 THEN PRINT "A SCROLL  
INSCRIBED WITH THE ANCIENT"  
10765 IF X = 7 THEN PRINT "MARTIAN  
LANGUAGE IS HERE."  
10770 IF X = 8 THEN PRINT "SOME  
GLITTERING STONES ARE HERE."  
10780 IF X = 9 THEN PRINT "A  
MYSTERIOUSLY HUMMING BOX IS HERE."
```

```

10790 IF X = 10 THEN PRINT "A LARGE,
      POLISHED SWORD IS HERE."
10800 IF X = 11 THEN PRINT "A BLEACHED
      SKULL IS HERE."
10810 IF X = 12 THEN PRINT "A SET OF
      BLUEPRINTS FOR AN ANCIENT"
10815 IF X = 12 THEN PRINT "MARTIAN
      PALACE IS HERE."
10820 Y = Y + 1: RETURN

```

The programming for the GET and DROP commands is listed in Listing 3.7, and flow-charted in Figure 3.6. While this routine is rather long, it is not complicated.

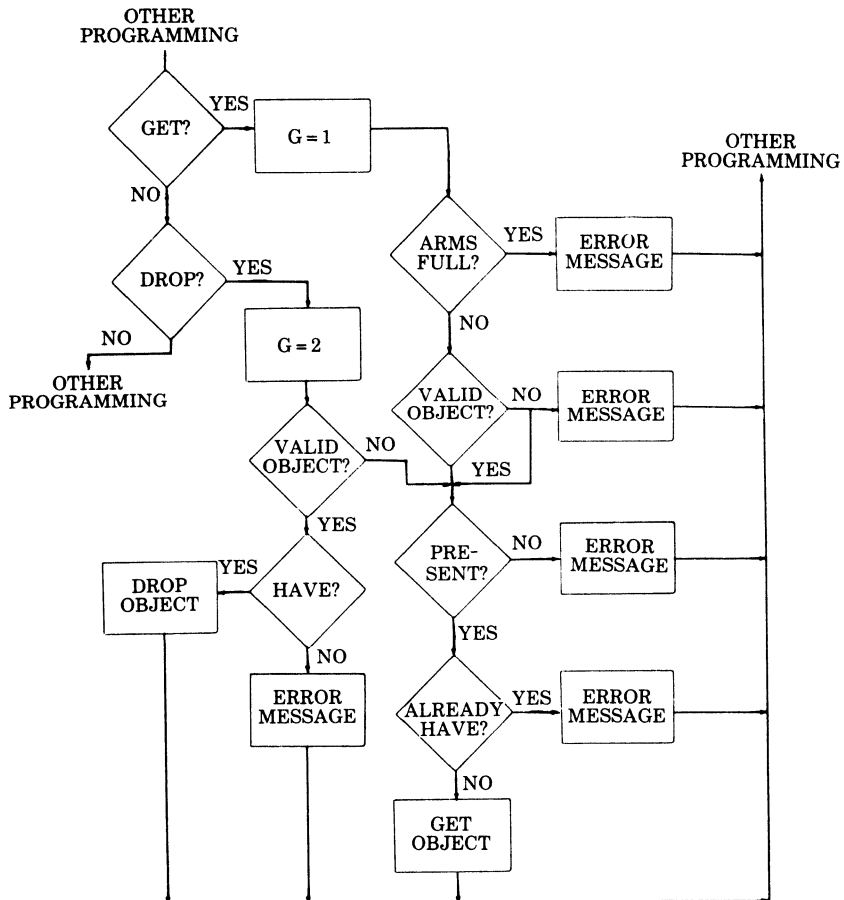


Figure 3.6 Flow-chart for GET/DROP.

Listing 3.7 Adding The GET and DROP Commands to the MARS Program.

```
500 REM*HOLD*
890 IF QX$ = "GET" THEN 6810
900 IF QX$ = "DRO" THEN 6920
6809 REM * GET/DROP *
6810 IF Q$ = "GET" THEN PRINT "GET
      WHAT?": GOTO 500
6815 IF Q$ = "GET ALL" THEN 9900
6820 IF Q$ = "OST" THEN 6870
6830 IF Q$ = "GET SICK" THEN 6880
6840 IF Q$ = "ENT" OR Q$ = "KED" THEN
      6890
6850 IF QY$ = "OWN" THEN 6900
6860 G = 1:Y = 0: FOR X = 1 TO 12:Y =
      Y + S(X) + J(X) + T(X): NEXT : IF
      Y > 17 THEN 9850
6865 GOTO 6930
6870 PRINT "I THINK YOU ALREADY ARE...":
      GOTO 500
6880 PRINT "THAT'S DISGUSTING!": GOTO
      500
6890 PRINT "SAME TO YOU, ";N$; "!": GOTO
      500
6900 PRINT "THIS IS NO TIME TO BOOGIE!":
      GOTO 500
6910 IF QY$ = "EAD" THEN 6890
6920 G = 2
6930 IF QY$ = "OOD" THEN U = 1: GOTO
      7500
6940 IF QY$ = "TLE" OR QY$ = "TER" THEN
      U = 2: GOTO 7500
6950 IF QY$ = "IFE" THEN U = 3: GOTO
      7500
6960 IF QY$ = "GUN" THEN U = 4: GOTO
      7500
6970 IF QY$ = "SER" THEN U = 5: GOTO
      7500
6980 IF QY$ = "OPE" OR QY$ = "OIL" THEN
      U=6: GOTO 7500
```

```
6990 IF QY$ = "AFT" THEN U = 7: GOTO
    7500
7000 IF QY$ = "GHT" THEN U = 8: GOTO
    7500
7010 IF QY$ = "IPE" THEN U = 9: GOTO
    7500
7020 IF RIGHT$ (Q$,4)= "INES" THEN
    U=10: GOTO 7500
7030 IF QY$ = "ASS" THEN U = 11: GOTO
    7500
7040 IF QY$ = "UIT" AND G = 2 THEN
    5070
7050 IF QY$ = "UIT" THEN U = 12: GOTO
    7500
7060 IF QY$ = "HOE" THEN U = 1: GOTO
    7600
7070 IF RIGHT$ (Q$,4) = "RING" THEN
    U = 2: GOTO 7600
7080 IF QY$ = "OCK" THEN U =3: GOTO
    7600
7090 IF QY$ = "RTS" THEN U = 4: GOTO
    7600
7100 IF QY$ = "IRT" OR QY$ = "LOT" THEN
    U = 5: GOTO 7600
7110 IF QY$ = "ONE" THEN U = 6: GOTO
    7600
7120 IF QY$ = "ICK" THEN U = 7: GOTO
    7600
7130 IF QY$ = "URN" THEN U = 8: GOTO
    7600
7140 IF QY$ = "WAD" OR QY$ = "GUM" THEN
    U = 9: GOTO 7600
7150 IF QY$ = "WER" THEN U = 10: GOTO
    7600
7160 IF QY$ = "FLY" THEN U = 11: GOTO
    7600
7170 IF RIGHT$ (Q$,4) = "HING" THEN
    U = 12: GOTO 7600
7180 IF QY$ = "PER" OR QY$ = "OWL" THEN
    U = 1: GOTO 7650
7190 IF QY$ = "INS" THEN U = 2: GOTO 7650
```



```
7200 IF QY$ = "ULE" THEN U = 3: GOTO
7650
7210 IF QY$ = "TUE" OR QY$ = "GOD" THEN
U = 4: GOTO 7650
7220 IF QY$ = "VER" OR QY$ = "CUP" THEN
U = 5: GOTO 7650
7230 IF QY$ = "ORB" THEN U = 6: GOTO
7650
7240 IF QY$ = "OLL" THEN U = 7: GOTO
7650
7250 IF RIGHT$ (Q$,4)= "ONES" THEN
U = 8: GOTO 7650
7260 IF QY$ = "BOX" THEN U = 9: GOTO
7650
7270 IF QY$ = "ORD" THEN U = 10: GOTO
7650
7280 IF QY$ = "ULL" THEN U = 11: GOTO
7650
7290 IF QY$ = "NTS" THEN U = 12: GOTO
7650
7300 X=LEN (Q$): IF X < 6 THEN 7350
7310 IF G=1 THEN X=X-4: GOTO 7320
7315 X = X - 5
7320 QY$=RIGHT$ (Q$,X)
7330 PRINT "SORRY ,";N$;", BUT I DON'T
SEE ANY":PRINT QY$;" HERE. "
7350 PRINT "PLEASE TRY TO KEEP YOUR
COMMANDS RATIONAL IN THE FUTURE."
7360 PRINT :DG=DG-3: GOTO 500
7500 IF G = 2 THEN 7550
7510 IF S(U) > 0 THEN 7540
7515 IF A(U) = L1 AND B(U) = L2 THEN
7520
7516 GOTO 7590
7520 S(U) = 1:A(U) = 0:B(U) = 0: PRINT
"OK": GOTO 500
7540 PRINT "YOU ALREADY HAVE IT!":DG
= DG-1: GOTO 500
7550 IF S(U) < 1 THEN 7580
7560 S(U)=0:A(U)=L1:B(U)=L2: PRINT
"OK": GOTO 500
```

```
7580 PRINT "YOU DON'T HAVE IT!":DG =  
    DG-1: GOTO 500  
7590 PRINT "IT'S NOT HERE, ";N$:DG =  
    DG-1: GOTO 500  
7600 IF G = 2 THEN 7630  
7610 IF J(U) > 0 THEN 7540  
7615 IFC(U) = L1 AND D(U) = L2 THEN  
    7620  
7617 GOTO 7590  
7620 J(U) = 1:C(U) = 0:D(U) = 0: PRINT  
    "OK": GOTO 500  
7630 IF J(U) < 1 THEN 7580  
7640 J(U) = 0:C(U) = L1:D(U) = L2: PRINT  
    "OK": GOTO 500  
7650 IF G = 2 THEN 7680  
7660 IF T(U) > 0 THEN 7540  
7665 IF E(U) = L1 AND F(U) = L2 THEN  
    7670  
7667 GOTO 7590  
7670 T(U) = 1:E(U) = 0:F(U) = 0: PRINT  
    "OK": GOTO 500  
7680 IF T(U) < 1 THEN 7580  
7690 T(U) = 0:E(U) = L1:F(U) = L2: PRINT  
    "OK": GOTO 500  
9850 PRINT "YOUR ARMS ARE FULL. YOU  
    CAN'T CARRY"  
9855 PRINT "ANYTHING MORE UNLESS  
    YOU DROP SOMETHING."  
9860 DG=DG-1: GOTO 500
```

For a GET command, G is set to a value of 1. This variable takes on a value of 2 for a DROP command.

An object must be specified in the command. Just saying GET doesn't mean much. You must use two words—for instance, GET KNIFE.

Lines 6820 through 6850, and 6910 check for wise-guy commands (such as GET LOST or DROP DEAD) and prints out appropriate messages (lines 6870 through 6900). They are not essential to the operation of the program, and may be eliminated or changed as you like.

QY\$ was set to the last three letters of the complete command (Q\$) in an earlier portion of the program. This string variable is now put to use to identify the object being acted on. Lines 6930 through 7290 check for a match with each of the 36 supply, junk, and treasure objects in the game. When a match is found the variable U is set to an appropriate value, and control is switched to a routine for acting on the appropriate arrays. The supply array routine begins at line 7500, the junk arrays at 7600, and the treasure arrays at 7650.

Some items may be identified by different names. For example, a player might try to pick up the coil of rope by entering GET COIL or GET ROPE. The program should recognize either name:

```
6980 IF QY$ = "OPE" OR QY$ = "OIL" THEN
      U = 6: GOTO 7500
```

You must also watch out for different item names that end in the same three letters, such as OLD MAGAZINES and GLITTERING STONES. To positively identify these items, you will need to check the last four letters of the original command (Q\$):

```
7020 IF RIGHT$ (Q$, 4) = "INES" THEN
      U = 10: GOTO 7500
7250 IF RIGHT$ (Q$, 4) = "ONES" THEN
      U = 8: GOTO 7650
```

If the player enters an item that is not included in the recognized object list, we will need to fall into an error routine. First we subtract GET or DROP from the original complete command to isolate the object name:

```
7300 X = LEN (Q$): IF X < 6 THEN 7350
7310 IF G = 1 THEN X = X-4: GOTO 7320
7315 X = X - 5
7320 QY$ = RIGHT$ (Q$, X)
```

For a GET command the first four letters are subtracted (the word GET, and the between word space). For a DROP command we delete the first five letters (D-R-O-P, and a space). You can now include the unrecognized object name in the error message:

```
7330 PRINT "SORRY ,";N$;", BUT I DON'T  
      SEE ANY":PRINT QY$;" HERE. "  
7350 PRINT "PLEASE TRY TO KEEP YOUR  
      COMMANDS RATIONAL IN THE FUTURE."  
7360 PRINT :DG=DG-3: GOTO 500
```

Notice that there is also a penalty to the current health rating (DG).

If the player enters GET DRUNK as his command, the computer will reply:

```
SORRY, DELTON, BUT I DON'T SEE ANY DRUNK HERE.  
Please try to keep your commands rational in the future.
```

Now, let's see what happens when a valid object name is entered with regard to the routine for the supply arrays. The junk and treasure routines work the same way.

Let's say the command is GET FOOD. There are three possible ways the computer may respond depending on current conditions:

- *FOOD is present
- *FOOD is not present
- *Character is already carrying FOOD

The S(X) array indicates which supply objects the player is already carrying; so check the appropriate supply location to determine if this is the case:

```
7510 IF S(U) > 0 THEN 7540  
7540 PRINT "YOU ALREADY HAVE IT!":DG  
      = DG-1: GOTO 500
```

If the character is not carrying the food, check the location arrays to see if the food and the player are in the same area:

```
7515 IF A(U) = L1 AND B(U) = L2 THEN  
      7520  
7516 GOTO 7590
```

If the specified object is not present, we once again have an error condition:

```
7590 PRINT "IT'S NOT HERE, ";N$:DG
    = DG-1: GOTO 500
```

Assuming the food is present, clear the appropriate location arrays and mark the appropriate point in the supplies array:

```
7520 S(U) = 1:A(U) = 0:B(U) = 0:
    PRINT "OK": GOTO 500
```

The displayed message OK confirms that the command has been obeyed.

If the command is DROP FOOD, there are only two possibilities. The character carrying the food can drop it; otherwise, an error message is displayed. Dropping an object is just like getting, except in reverse. The player's current coordinates are placed into the appropriate object location arrays, and the object is deleted from the supplies carried array:

```
7500 IF G = 2 THEN 7550
7550 IF S(U) < 1 THEN 7580
7560 S(U) = 0:A(U) = L1:B(U) = L2:
    PRINT "OK": GOTO 500
7580 PRINT "YOU DON'T HAVE IT!":DG =
    DG-1: GOTO 500
```

To make the game a little harder, allow the explorer to carry only up to 17 items at a time. This routine will only be appropriate for GET commands. The player can DROP as many items as he is carrying:

```
6860 G = 1:Y = 0: FOR X = 1 TO 12:Y =
    Y + S(X) + J(X) + T(X): NEXT: IF
    Y > 17 THEN 9850
9850 PRINT "YOUR ARMS ARE FULL. YOU
    CAN'T CARRY"
9855 PRINT "ANYTHING MORE UNLESS
    YOU DROP SOMETHING."
9860 DG=DG-1: GOTO 500
```

At the beginning of each game, the explorer starts out aboard the rocket ship with all 12 supplies present. He will probably want to GET most of them. Entering 12 separate commands is a bit of a

nuisance, so allow a GET ALL command to gather the supplies aboard the rocket ship but under no other conditions. The programming needed to accomplish this is in Listing 3.8.

Listing 3.8 The GET ALL Routine for the MARS Program.

```
6815 IF Q$ = "GET ALL" THEN 9900
9899 REM * GET ALL *
9900 IF L1 = R1 AND L2 = R2 THEN 9920
9910 GOTO 970
9920 FOR X = 1 TO 12: IF A(X) = L1 AND
      B(X) = L2 THEN 9950
9930 NEXT : PRINT "SUPPLIES GATHERED. ":
      GOTO 500
9950 A(X) = 0:B(X) = 0:S(X) = 1: GOTO
      9930
```

Two of the supplies are of special significance. As stated earlier, the current coordinates should be displayed only when the player is carrying the COMPASS. You can easily add line 410 to include an IF...THEN... test:

```
410 IF S(11) = 0 THEN PRINT "??:?":
      GOTO 420
```

It is logical to assume that the explorer can only survive outside his rocket ship if he is wearing his space suit. Leaving the ship without the suit, or dropping the suit, becomes fatal by adding the steps shown in Listing 3.9.

Listing 3.9 Special Programming for the Compass and the Spacesuit in the Game of MARS.

```
410 IF S(11) = 0 THEN PRINT "??:?":
      GOTO 420
415 PRINT L1;" ";L2
420 IF L1 = R1 AND L2 = R2 THEN 430
425 IF S(12) = 0 THEN 5000
430 IF L1 = R1 AND L2 = R2 THEN PRINT
      "YOU ARE SAFELY ABOARD YOUR ROCKET
      SHIP."
```

```

435 IF L1 = R1 AND L2 = R2 THEN DG = DG
    + 3
5000 PRINT "YOU ARE OUTSIDE WITHOUT
    YOUR SPACESUIT!": PRINT
5010 FOR X = 1 TO 4:Y = INT (RND (1) *
    5 + 1):Z = INT (RND (1) * 75 + 1)
    + 25: FOR ZZ = 1 TO Z: NEXT
5020 IF Y = 1 THEN PRINT "* GASP *",
5030 IF Y = 2 THEN PRINT "* CHOKE *",
5040 IF Y = 3 THEN PRINT "* PANT-PANT *",
5050 IF Y = 4 THEN PRINT "* WHEEZE *",
5060 NEXT X: PRINT
7040 IF QY$ = "UIT" AND G = 2 THEN 5070

```

Also notice that being aboard the rocket ship is beneficial, in that 3 is added to the current health rating (DG) of the character.

INVENTORY

Since it is easy to lose track as the game goes on of what your explorer is carrying, add a command to display a list of the items being carried. Call this command INVENTORY, or INV for short.

To program in the INVENTORY function to the game of MARS, see Listing 3.10. In this simple routine each object carried array location is checked. If the value is 1, the name of the appropriate object is displayed.

Various objects are listed in a random order to help block attempts to distinguish between junk and treasure items based on their displayed position.

Listing 3.10 Adding the INVENTORY Command to the MARS Program.

```

840 IF QX$ = "INV" THEN GOSUB 11300:
    GOTO 700
11299 REM *INVENTORY *
11300 PRINT : PRINT "YOU ARE NOW
    CARRYING ---"
11310 IF S(1) = 1 THEN PRINT "FOOD"
11320 IF J(2) = 1 THEN PRINT "ORNATE
    RING"
11330 IF T(3) = 1 THEN PRINT "FOSSILIZED
    SLIDE RULE"

```

```
11340 IF J(4) = 1 THEN PRINT "FOSSILIZED
      UNDERSHORTS"
11350 IF S(5) = 1 THEN PRINT "LASER"
11360 IF J(6) = 1 THEN PRINT "OLD BONE"
11370 IF T(7) = 1 THEN PRINT "SCROLL"
11380 IF J(8) = 1 THEN PRINT "URN"
11390 IF S(9) = 1 THEN PRINT "METAL
      PIPE"
11400 IF J(10) = 1 THEN PRINT "FLOWER"
11410 IF T(11) = 1 THEN PRINT "SKULL"
11420 IF J(12) = 1 THEN PRINT "SLIMY
      THING (?) "
11430 IF S(11) = 1 THEN PRINT "COMPASS"
11440 IF T(10) = 1 THEN PRINT "LARGE
      SWORD"
11450 IF J(9) = 1 THEN PRINT "PETRIFIED
      WAD OF BUBBLE GUM"
11460 IF T(8) = 1 THEN PRINT "GLITTERING
      STONES"
11470 IF S(7) = 1 THEN PRINT "INFLATABLE
      RAFT"
11480 IF T(6) = 1 THEN PRINT "GLASS ORB"
11485 INPUT "PLEASE PRESS RETURN TO
      CONTINUE";Q$
11490 IF J(5) = 1 THEN PRINT "CLOT OF
      DIRT"
11500 IF T(4) = 1 THEN PRINT "STATUE OF
      MARTIAN GOD"
11510 IF S(3) = 1 THEN PRINT "KNIFE"
11520 IF T(2) = 1 THEN PRINT "GOLD
      COINS"
11530 IF J(1) = 1 THEN PRINT "OLD
      SHOE"
11540 IF T(1) = 1 THEN PRINT "COPPER
      BOWL"
11550 IF S(2) = 1 THEN PRINT "BOTTLE
      OF WATER"
11560 IF S(2) = 2 THEN PRINT "EMPTY
      BOTTLE"
11570 IF J(3) = 1 THEN PRINT "ROCK"
11580 IF S(4) = 1 THEN PRINT "GUN"
11590 IF T(5) = 1 THEN PRINT "SILVER CUP"
```



```

11600 IF S(6) = 1 THEN PRINT "COIL
      OF ROPE"
11610 IF J(7) = 1 THEN PRINT
      "SHARPENED STICK"
11620 IF S(8) = 1 THEN PRINT
      "FLASHLIGHT"
11630 IF T(9) = 1 THEN PRINT
      "MYSTERIOUSLY HUMMING BOX"
11640 IF S(10) = 1 THEN PRINT "OLD
      MAGAZINES"
11650 IF J(11) = 1 THEN PRINT
      "BUTTERFLY"
11660 IF T(12) = 1 THEN PRINT
      "BLUEPRINTS"
11670 IF S(12) = 1 THEN PRINT
      "SPACESUIT"
11680 Z = 0: FOR Y = 1 TO 12: Z = Z +
      S(Y) + J(Y) + T(Y): NEXT Y: IF Z =
      0 THEN PRINT "NOTHING"
11690 PRINT : RETURN

```

BLAST OFF

As you know the way to win the game is to bring as many Martian treasures as possible aboard the rocket ship and blast off for Earth. Obviously, you need a blast off command.

The game of MARS ends when either the explorer gets killed, or the player uses the BLAST OFF command.

The player should only be able to BLAST OFF aboard the rocket ship. If this command is entered at any other location, a sarcastic message is displayed (line 9970).

When the player chooses to BLAST OFF aboard the rocket ship, the final score will be displayed. A score of better than 75 will result in a congratulatory message from the computer. If the score is below 25, the computer will express dissatisfaction.

The BLAST OFF routine is Listing 3.11.

Listing 3.11 Adding the BLAST OFF Command to the Game of MARS.

```

9960 IF QX$ = "BLA" THEN 9960
9960 IF L1 = R1 AND L2 = R2 THEN 9980

```

```

9970 PRINT "YOU DON'T HAVE JET
      PROPULSION ENGINES IN"
9975 PRINT "YOUR SHOES!":DG
      = DG - 2: GOTO 500
9980 FOR X = 1 TO 100: PRINT " * ";: FOR
      Y = 1 TO 55: NEXT : NEXT
9985 PRINT : PRINT : GOSUB 10400
9990 PRINT "YOUR SCORE WAS ";SC: IF SC >
      75 THEN PRINT "FANTASTIC
      WORK, ";N$;"!"
9995 IF SC < 25 THEN PRINT "I'M NOT TOO
      IMPRESSED BY YOUR"
9996 IF SC < 25 THEN PRINT
      "PERFORMANCE ..."
9997 GOTO 5092

```

THE EAT AND DRINK COMMANDS

Since the supplies include food and a bottle of water, include EAT and DRINK commands. Programming for these commands is flow-charted in Figure 3.7 and is in Listing 3.12.

EATING or DRINKING boosts the character's current health rating (DG), but will not let it surpass the maximum health rating (DX).

In the next chapter, add the monsters. Some dead monsters may be edible, so the EAT command must consist of two words to identify what is to be eaten.

To EAT in front of certain live monsters will have special results, so we will hold a space for the appropriate IF...THEN... tests to be added later:

```

800 IF QX$= "EAT" THEN 5300
5300 REM*HOLD FOR MONSTERS*
5340 X=LEN (Q$): IF X < 5 THEN 5590
5590 PRINT "EAT WHAT?": DG=DG-.25:
      GOTO 500

```

If EAT FOOD is specified, the computer must check to see if the explorer is actually carrying any food (S(1)):

```

5310 IF QY$= "OOD" THEN 5420
5420 IF S(1)=1 THEN 5450
5430 PRINT "YOU DON'T HAVE ANY!"
5440 DG=DG-1: GOTO 500

```

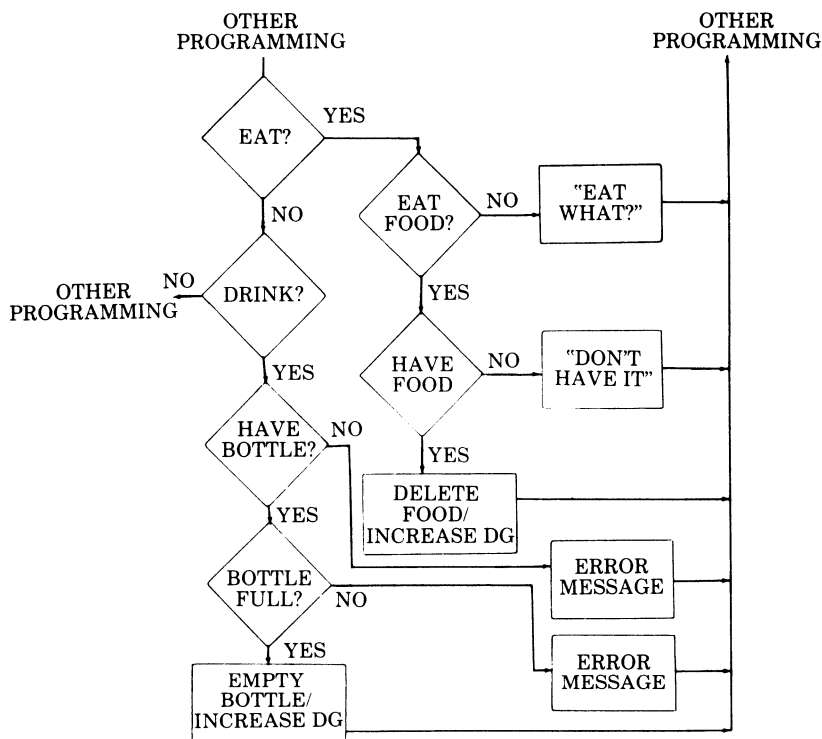


Figure 3.7 Flow-chart for EAT/DRINK.

Listing 3.12 Adding the EAT and DRINK Commands to the MARS Program.

```

800 IF QX$ = "EAT" THEN 5300
810 IF QX$ = "DRI" THEN 5700
5299 REM * EAT *
5300 REM*HOLD FOR MONSTERS*
5310 IF QY$ = "OOD" THEN 5420
5340 X = LEN (Q$): IF X < 5 THEN 5590
5350 X = INT (RND (1) * 6 +1):DG = DG
    - .5: IF X = 1 THEN PRINT "EAT
    WHAT???"
5360 IF X = 2 THEN PRINT "-ER- NO,
    THANK YOU..."
5370 IF X = 3 THEN PRINT "ARE YOU
    NUTS?"
  
```

```
5380 IF X = 4 THEN PRINT "THAT HUNGRY,  
    I'M NOT!"  
5390 IF X = 5 THEN PRINT "YUK!"  
5400 IF X = 6 THEN PRINT "HAS ANYONE  
    EVER TOLD YOU THAT YOU HAVE  
    WEIRD TASTES?"  
5410 GOTO 500  
5420 IF S(1) = 1 THEN 5450  
5430 PRINT "YOU DON'T HAVE ANY!"  
5440 DG = DG - 1: GOTO 500  
5450 PRINT " ", "BURP": PRINT  
5460 DG = DG + (DX * .25): IF DG > DX  
    THEN DG = DX  
5470 S(1) = 0: GOTO 500  
5590 PRINT "EAT WHT?": DG = DG - .25  
    : GOTO 500  
5699 REM * DRINK *  
5700 REM*HOLD FOR RIVER*  
5710 IF S(2) = 1 THEN 5740  
5720 IF S(2) = 2 THEN 5800  
5730 PRINT "THERE IS NOTHING HERE TO  
    DRINK, "; N$: DG = DG - .5: GOTO 500  
5740 PRINT " ", : FOR X = 1 TO 3: FOR  
    Y = 1 TO 85: NEXT Y  
5750 PRINT "* GLUG * ", : NEXT X: PRINT  
5760 GOSUB 10000  
5770 PRINT " ", "AHHHH!": DG = DG + (DX  
    * .15)  
5780 IF DG > DX THEN DG = DX  
5790 S(2) = 2: GOTO 500  
5800 PRINT "YOUR WATER BOTTLE IS  
    EMPTY": DG = DG - .4: GOTO 500  
11560 IF S(2)=2 THEN PRINT "EMPTY BOTTLE"
```

If the explorer does have the food, it will be deleted from the array since you can't eat the same food twice, and the current health rating (DG) will be increased by up to 25% of the maximum (DX), without exceeding it:

```
5450 PRINT " ", "BURP": PRINT  
5460 DG=DG+(DX*.25): IF DG > DX THEN DG=DX  
5470 S(1)=0: GOTO 500
```

Lines 5350 through 5410 display one of six randomly selected error messages in case the player tries to have his character EAT something the computer does not recognize as edible (for example, EAT FOOT). At this point only food is recognized, but a few other items will be added in the next chapter.

The DRINK command is basically similar to the EAT command. The command is only valid if the explorer has the bottle of water, or is at the river—to be added in the next chapter.

After drinking, the water should be gone, but the bottle would logically still be there. By inserting a value of 2 into array location S(2), you can replace the bottle of water with an empty bottle. Be sure to add this line to the INVENTORY subroutine:

```
11560 IF S(2)=2 THEN PRINT "EMPTY BOTTLE"
```

Drinking from an empty bottle is, of course, not permitted. In the next chapter, you will add a command to refill the bottle at the river.

There is another magical way to refill the bottle. See if you can figure out how to refill your water bottle without a river.

**Table 3.4 Routines and Subroutines for the MARS Program
Presented in Chapter 3.**

Routines

1-10	initialization
20-170	variable preset
180-380	set character attributes
400-410	display current location
420-430	aboard rocket ship
440-480	check for objects present
650-660	health check
700-980	main command input and check
50000-5060	character dead/lose game
5100-5170	directional moves
5180-5210	CRY
5300-5590	EAT
5700-5800	DRINK
6550-6590	PRAY
6810-7690	GET/DROP
9820-9840	WAIT

9850-9860	arms full
9900-9950	GET ALL
9960-9995	BLAST OFF

Subroutines

10000	timing delay loop
10010	instructions (incomplete)
10400-10440	score calculation
10450-10565	supplies present
10570-10690	junk items present
10700-10820	treasure items present
11150-11260	DIAGNOSIS
11300-11690	INVENTORY
11700-11780	error messages
11800-11870	HELP

Table 3.5 Variables Used in the MARS Program So Far.

AX	character's aim rating
DG	character's current health rating
DX	character's maximum health rating
G	GET or DROP
K	monster previously encountered
L1,L2	character's current location
L3,L4	character's previous location
Q	monster/obstacle present
PX	character's power rating
R1,R2	rocket shop location
SC	score
SV	monster kill score

SUMMARY

You now have a complete game of sorts, albeit a rather dull one at this point because it lacks real obstacles. In the next chapter programming will begin to liven up.

But before you add the monsters and other complications to the program, you should thoroughly test the partial program written so far. Try each of the commands several times. Enter bad commands to ensure that the error-trapping routines work properly. At this point, 17 commands are recognized. They are as follows:

N—move north
S—move south

E—move east
W—move west
SCORE—display current score
DIA—display diagnosis (report on current health rating)
HELP—display list of possible commands
WAIT—pause and increase current health rating
LOOK—re-display current coordinates
GET xxx—pick up an object (2 words required)
DROP xxx—set down an object (2 words required)
INV—inventory objects currently being carried
EAT xxx—2 words required
DRINK
CRY
PRAY
BLAST OFF—return to Earth; display final score and end the game

Once you are sure everything in the program so far works correctly, move on to the next chapter and complicate the poor explorer's life with monsters and geographic obstacles.

Chapter 4

Complicating the Game

The *heart and soul* of a good adventure game is the monsters and obstacles that plague the player in his quest.

This chapter adds disasters, setbacks, and even strokes of fortune to the MARS program begun in the last chapter. Here you will create creatures and landmarks.

MAPPING MONSTERS

The variable preset routine of Listing 3.2 planted the obstacles in the main location map (LC(x,y)) with lines 40 to 60. Although the monsters and obstacles are in the program, the player can't see them yet.

First, add a display that will tell the player what obstacles, if any, he currently faces. In line 400, the variable Q was set to equal the value stored in the main map for the player's current location. Use the value of Q to determine whether anything is at the current map location.

A temporary routine for naming each obstacle as it is encountered is included in Listing 4.1. Most of the lines will be changed as you move on to more advanced programming.

Listing 4.1 Routine to Display the Obstacles in the MARS Program (Temporary Routine).

```
499 REM*CHECK FOR MONSTERS*
500 EX(L1,L2)=Q
510 IF Q=1 THEN PRINT "SQUEANLY SERPENT"
520 IF Q=2 THEN PRINT "GHOST"
530 IF Q=3 THEN PRINT "BRINCHLEY BEAST"
```



```

540 IF Q=4 THEN PRINT "KUFU"
550 IF Q=5 THEN PRINT "GRIMPH"
560 IF Q=6 THEN PRINT "PUROFOLEE"
570 IF Q=7 THEN PRINT "RIVER"
580 IF Q=8 THEN PRINT "MOUNTAIN"
590 IF Q=9 THEN PRINT "RAVINE"
600 IF Q=10 THEN PRINT "MARSQUAKE"
610 IF Q=11 THEN PRINT "STORM"
620 IF Q=12 THEN PRINT "FUNNY COLORED SKY"

```

Add this routine to those from Chapter 3 and run a sample. Scatter all the obstacles throughout the map area. Some locations will be blank. Be sure that no monsters appear in the rocketship's location.

Next add a map display routine as shown in Listing 4.2, with the flow-chart shown in Figure 4.1.

Listing 4.2 Adding the MAP Command to the MARS Program.

```

950 IF QX$ = "MAP" THEN GOSUB 12400 :
      GOTO 700
12399 REM * MAP *
12400 PRINT CHR$(147) : PRINT : PRINT
12410 FOR X = 1 TO 10: PRINT " ";: FOR
      Y = 1 TO 10
12415 IF L1 = X AND L2 = Y THEN PRINT
      "+ ";: GOTO 12450
12420 Z = EX(X,Y): IF Z < 1 OR Z > 9
      THEN 12440
12430 ON Z GOTO 12500,12510,12520,
      12530,12540,12550,12560,12570,12580
12440 IF Z = 20 THEN PRINT "* ";: GOTO
      12450
12445 PRINT ". ";
12450 NEXT : PRINT : NEXT : PRINT
12460 PRINT "+= YOU, *= SHIP, S=
      SQUEANLEY SERPENT,"
12465 PRINT "B= BRINCHLEY
      BEAST, K= KUFU, G=GRIMPH,"
12470 PRINT "P=PUROFOLEE, R= RIVER, M=
      MOUNTAIN,"
12475 PRINT "V= RAVINE":PRINT"  PRESS
      'RETURN' TO CONTINUE GAME  ";

```

```
12490 INPUT Q$: RETURN
12500 PRINT "S ";; GOTO 12450
12510 PRINT ". ";; GOTO 12450
12520 PRINT "B ";; GOTO 12450
12530 PRINT "K ";; GOTO 12450
12540 PRINT "G ";; GOTO 12450
12550 PRINT "P ";; GOTO 12450
12560 PRINT "R ";; GOTO 12450
12570 PRINT "M ";; GOTO 12450
12580 PRINT "V ";; GOTO 12450
```

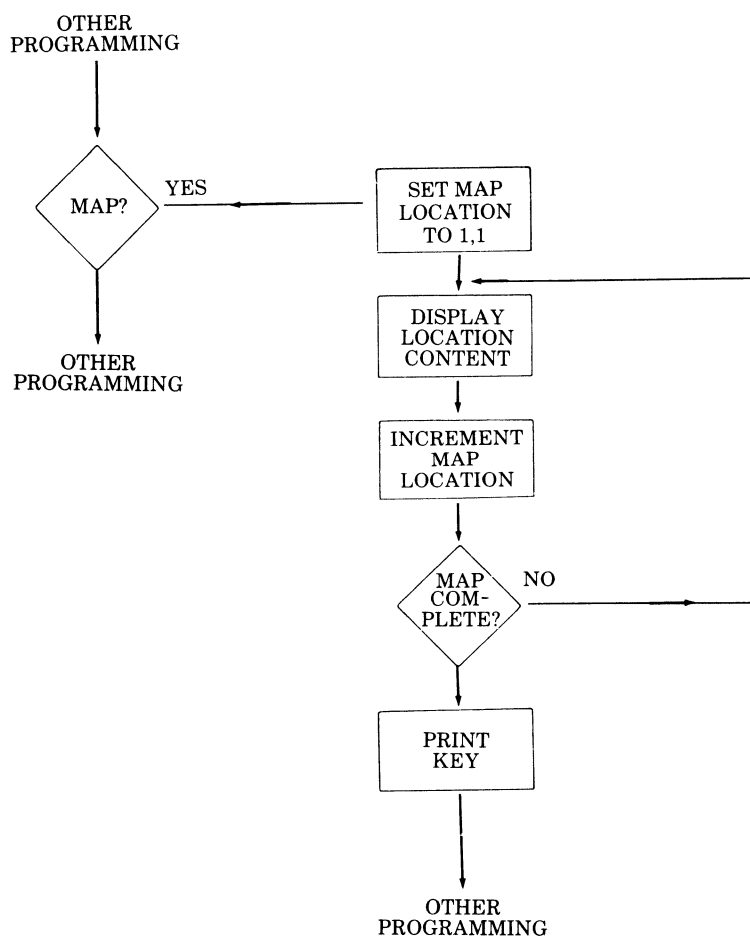


Figure 4.1 Flow-chart for MAP Routine.

Notice that the explored map (EX(x,y)) is the one displayed – not the complete map (LC(x,y)). Only those obstacles that the player has already encountered are displayed.

A typical map display is shown in Figure 4.2. Notice that the Marsquake, storm, and funny-colored sky are not displayed. Neither is the ghost displayed on the map since each ghost appears only once, and then vanishes, leaving its space blank.

```

.   .   .   .   .   .   .   .   .   .
.   *   .   .   .   .   .   .   .   .
K   B   .   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .   .   .
G   M   .   .   .   .   .   .   .   .
P   R   .   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .   .   .
.   r   .   .   .   .   +   .   .   M
.   .   .   .   .   .   .   .   .   .

```

Figure 4.2 Typical Map for the Game of MARS.

SQUEANLY SERPENT

Just to wander about and be told monsters are present isn't too exciting. Later additions to the command sequence will allow the player more direct interaction with some of the obstacles.

For now, write expanded display routines so each type of monster begins to show a unique personality.

Start with the Squeanly Serpent, which appears when Q has a value of 1. First, change the identification line to call up a subroutine rather than just print out the creature's name:

```
510 IF Q = 1 THEN GOSUB 10950
```

The Squeanly Serpent is a fairly simple beast that bites the explorer who encounters him and that's that. Of course, this is detrimental to the character's current health rating.

A player will have no way to kill, or deal directly with a Squeanly Serpent. When he encounters one of these nasty reptiles, the only thing he can do is to move on to a new location. Listing 4.3 has the complete Squeanly Serpent subroutine.

Listing 4.3 Squeanly Serpent.

```
510 IF Q = 1 THEN GOSUB 10950
10949 REM * SERPENT *
10950 PRINT "DARN! YOU WERE JUST
      BITTEN BY A SQUEANLEY
      SERPENT!"
10960 DG = DG -5: RETURN
```

GHOST

At some locations (when Q=2), a ghost of an ancient Martian appears:

```
520 IF Q=2 THEN GOSUB 11000
```

Sometimes the ghost is beneficial, and other times it will be neutral. There is no need to make everything in the game to the player's disadvantage.

Just a displayed message announces the ghost's appearance:

```
11000 PRINT "A GHOST OF AN ANCIENT
      MARTIAN SUDDENLY APPEARS
      BEFORE YOU!"
```

Then a pair of random coordinates are selected and the contents of that location is stored as Z:

```
11010 X = INT (RND (1)*10+1):
      Y =INT (RND (1)*10+1): Z =
      LC(L1,L2)
```

If Z has a value of 1, 3, 4, 5, or 6 the ghost will announce its selected coordinates and tell what type of monster occupies that location:

```
11040 PRINT "'LO,' SAYETH THE GHOST,
      'AT ": PRINT L1;" ";L2;" YOU
      SHALL FIND A ";
```

```

11050 IF Z=1 THEN PRINT "SQUEANLEY
      SERPENT"
11060 IF Z=3 THEN PRINT "BRINCHLEY BEAST"
11070 IF Z=4 THEN PRINT "KUFU"
11080 IF Z=5 THEN PRINT "GRIMPH"
11090 IF Z=6 THEN PRINT "PUROFOLEE"

```

When this information is recorded in the explored map array (EX(x,y)), the ghost vanishes, being erased from both the main and the explored maps. The subroutine ends here:

```

11100 EX(L1,L2)=Z
11110 PRINT "THE GHOST VANISHES INTO
      THIN AIR!":RETURN

```

Since the ghost only reveals the location of certain types of obstacles, make some provision if the randomly selected coordinates do not contain one of these monsters.

If the selected location has a value greater than 6, a new set of coordinates will be selected:

```

11030 IF Z > 6 THEN 11010

```

The map location may also take a value of 2, or 0 (or negative numbers, which will be introduced later). If these values are encountered, you can loop around to select a new set of coordinates. However, the computer may take a long time to select a valid set of coordinates, especially late in a game when many of the monsters are dead. You need an alternative routine to prevent the program from latching up.

If the selected coordinates are the location of another ghost (Z=2), or nothing (Z=0) or a dead monster ($-z < 0$), the ghost will just say "BOO!" and vanish:

```

11020 IF Z < 1 OR Z=2 THEN 11120
11120 PRINT " ", "BOO!":PRINT: GOTO 11110

```

Complete programming for the ghost is listed in Listing 4.4. A flow-chart is shown in Figure 4.3.

As with the Squeanly Serpent, there is no way the player can interact with a ghost. The ghost appears, says its bit, then vanishes

permanently from the game. You may encounter more than one ghost.

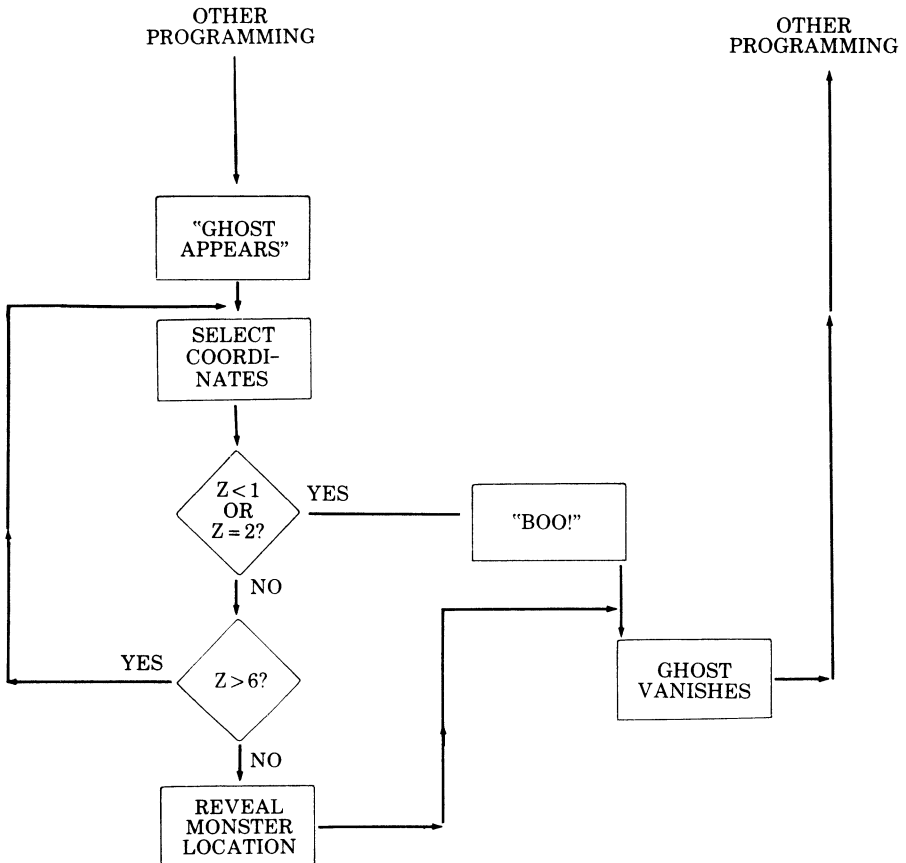


Figure 4.3 Flow-chart for Ghost Routine.

Listing 4.4 Ghost.

```

520 IF Q = 2 THEN GOSUB 11000
11000 PRINT "A GHOST OF AN ANCIENT
      MARTIAN SUDDENLY APPEARS
      BEFORE YOU!"
11010 X = INT (RND (1)*10+1):
      Y = INT (RND (1)*10+1): Z =
      LC(L1,L2)
  
```

```

11020 IF Z < 1 OR Z=2 THEN 11120
11030 IF Z > 7 THEN 11010
11040 PRINT "'LO,' SAYETH THE GHOST,
      'AT ": PRINT L1; ":"; L2; " YOU
      SHALL FIND A ";
11050 IF Z=1 THEN PRINT "SQUEANLEY
      SERPENT"
11060 IF Z=3 THEN PRINT "BRINCHLEY BEAST"
11070 IF Z=4 THEN PRINT "KUFU"
11080 IF Z=5 THEN PRINT "GRIMPH"
11090 IF Z=6 THEN PRINT "PUROFOLEE"
11100 EX(L1,L2)=Z
11110 PRINT "THE GHOST VANISHES INTO
      THIN AIR!":RETURN
11120 PRINT " ", "BOO!":PRINT: GOTO 11110

```

BRINCHLEY BEAST

The Brinchley Beast is the first monster with which the player can interact and fight:

```
530 IF Q = 3 THEN 9300
```

When facing a Brinchley Beast, certain commands will have special results that will be dealt with later. For now, concentrate on the initial encounter.

One way to give your game creatures personality is to simulate their cries on the display screen. The Brinchley Beast gets its name from its distinctive cry of "brinch-LEY!!!"

The cry will be displayed only once when the creature is first encountered. The variable K (which is set to 0 on each location move) will be given a value of 1 to indicate the Beast has already been encountered. When K equals 1, the cry is skipped, and just a message stating that a Brinchley Beast is there will be displayed.

The first time a player encounters a Brinchley Beast, another variable (TB) is set to 0. This variable indicates whether or not the explorer has touched the Beast. This is important when you add the special commands later.

Listing 4.5 shows the complete encounter routine for a Brinchley Beast.

Listing 4.5 Brinchley Beast.

```
530 IF Q = 3 THEN 9300
9300 IF K = 1 THEN 9330
9310 K = 1:TB = 0: PRINT " ", "BRINCH-";:
      GOSUB 10000
9320 PRINT "LEY!!!": PRINT :GOSUB 10000
9330 PRINT "A BRINCHLEY BEAST IS HERE.":
      GOTO 650
```

KUFU

The next monster encountered (Q=4) is the Kufu, a much nastier creature than the Brinchley Beast as indicated by the message which is displayed when a Kufu is first encountered:

A hungry Kufu blocks your path!

It salivates in foul anticipation of the meal to come (lines 9360 through 9380).

The variable K, set to a value of 2, indicates that this Kufu has already been encountered. On subsequent encounters, a different message is displayed:

```
9350 IF K = 2 THEN 9400
9400 PRINT "A KUFU IS HERE."
```

In either case, a random number from one to 10 is selected. If this number is greater than 6 (a 40 percent chance), the Kufu will attack, biting the explorer in one of seven randomly selected places. Depending on where the character is bitten, points are subtracted from his current health rating (DG). A neck bite is much more serious than an arm bite. If the Kufu bites the explorer's air hose, the character dies and the game ends:

```
9410 X = INT(RND (1)*10+1): IF
      X > 6 THEN 6670
6670 PRINT "IT BITES YOUR ";:X =
      INT (RND (1)*7+1)
```



```

6680 IF X = 1 THEN PRINT "ARM!";
      DG = DG-5
6690 IF X = 2 THEN PRINT "LEG!";
      DG = DG-7
6700 IF X = 3 THEN PRINT "STOMACH!";
      DG = DG-50
6710 IF X = 4 THEN PRINT "NECK!";
      DG = DG-75
6720 IF X = 5 THEN PRINT "AIR HOSE!";
      GOTO 5070
6730 IF X = 6 THEN PRINT "NOSE!";
      DG = DG-20
6740 IF X = 7 THEN PRINT "POSTERIOR!";
      DG = DG-30

```

Whether or not the Kufu attacks, program control is jumped back to line 650 for a health check. Even a bite on the arm can be fatal if the character is in weak enough condition ($DG < 5$).

Soon you will add commands for dealing with the Kufu. Listing 4.6 shows the programming for the Kufu encounter.

Listing 4.6 Kufu.

```

540 IF Q = 4 THEN 9350
9350 IF K = 2 THEN 9400
9360 PRINT "A HUNGRY KUFU BLOCKS YOUR
      PATH!"; GOSUB 10000
9370 PRINT : PRINT "IT SALIVATES IN FOUL
      ANTICIPATION OF THE MEAL TO COME!";
      PRINT
9380 K = 2: GOTO 9410
9400 PRINT "A KUFU IS HERE."
9410 X = INT (RND (1) * 10 + 1): IF X >
      6 THEN 6670
9420 GOTO 650
6670 PRINT "IT BITES YOUR "; X =
      INT (RND (1)*7+1)
6680 IF X = 1 THEN PRINT "ARM!";
      DG = DG-5
6690 IF X = 2 THEN PRINT "LEG!";
      DG = DG-7

```

```
6700 IF X = 3 THEN PRINT "STOMACH!":  
    DG = DG-50  
6710 IF X = 4 THEN PRINT "NECK!":  
    DG = DG-75  
6720 IF X = 5 THEN PRINT "AIR HOSE!":  
    GOTO 5070  
6730 IF X = 6 THEN PRINT "NOSE!":  
    DG = DG-20  
6740 IF X = 7 THEN PRINT "POSTERIOR!":  
    DG = DG-30  
6750 GOTO 650
```

GRIMPH

The next monster ($Q=5$), in Listing 4.7, is called a Grimp, the nastiest monster in the game of MARS. When we add new commands, the Grimp will be very difficult to fight or escape from.

Listing 4.7 Grimp.

```
550 IF Q = 5 THEN 9450  
9110 PRINT "THE GRIMP BREAKS YOU ";  
9120 X = INT (RND (1) * 10 + 1): IF  
    X = 1 AND S(11) = 0 THEN 9120  
9130 IF X = 1 THEN PRINT "COMPASS!"  
    :S(11) = 0  
9140 IF X = 2 THEN PRINT "ARM!": DG =  
    DG - 15  
9150 IF X = 3 THEN PRINT "LEG!": DG =  
    DG - 20  
9160 IF X = 4 THEN PRINT "NECK!": DG =  
    DG - 70  
9170 IF X = 5 THEN PRINT "THUMBNAIL!":  
    DG = DG - 2  
9180 IF X = 6 THEN PRINT "NOSE!": DG =  
    DG - 10  
9190 IF X = 7 THEN PRINT "BIG TOE!": DG  
    = DG - 3  
9200 IF X = 8 THEN PRINT "BACK!": DG =  
    DG - 65
```

```

9210 IF X = 9 THEN PRINT "FINGER!": DG
      = DG - 3
9220 IF X = 10 THEN PRINT "SKULL!": DG
      = DG - 75
9230 GOTO 650
9450 PRINT "A GRIMPH IS HERE.":X = INT
      (RND (1) * 10 + 1): IF X > 3 THEN
      9110
9460 GOTO 650

```

When a Grimp is encountered, there is a 70 percent chance that it will attack:

```

9450 PRINT "A GRIMPH IS HERE.":X = INT
      (RND (1)*10+1): IF X > 3 THEN 9100

```

A Grimp, when it attacks, will break either some part of the explorer's anatomy, whether it be a thumbnail or a skull. He might also break the compass which is necessary for the current coordinates to be displayed.

Obviously, the character must be carrying the compass in order for the Grimp to break it, so an extra IF...THEN... statement is added so that if the compass is selected, but is not there (S(11)=0) a new attack number will be selected:

```

9120 X = INT (RND (1)*10+1): IF X = 1
      AND S(11)=0 THEN 9120

```

With or without an attack, an encounter with a Grimp ends with a health check (line 650).

PUROFOLEE

The Purofolee is the last of the monsters a player can encounter. This creature appears when Q has a value of 6. Listing 4.8 shows the programming for an encounter. K is set to a value of 3 after the first encounter.

On the first encounter, the Purofolee's cry of "Gibble! Gibble! Gibble!" is heard (displayed) followed by "A wild Purofolee hops into view."

On later encounters the computer simply reminds you that "A Purofolee is here." and the Purofolee inquires about your intentions:

"PUROFOLEE: Gobble?"

As with other monsters, several new commands will soon be added to the program for dealing with a Purofolee.

Listing 4.8 Purofolee.

```
560 IF Q = 6 THEN 9500
9500 IF K = 3 THEN 9550
9510 GOSUB 10000: FOR X = 1 TO 3:Z =
      INT (RND (1) * 25 + 1) + 1: FOR
      Y = 1 TO Z
9520 PRINT " ";: NEXT Y: PRINT
      "GIBBLE! ";: NEXT X
9530 PRINT : PRINT : PRINT "A WILD
      PUROFOLEE HOPS INTO VIEW!":K = 3
9540 GOTO 650
9550 PRINT "A PUROFOLEE IS HERE.":
      PRINT
9560 PRINT "PUROFOLEE:      GIBBLE?":
      PRINT
9570 GOTO 650
```

RIVER

Now add natural obstacles or landmarks to the array of monsters. The first of these is a river (Q=7), programmed in Listing 4.9.

Listing 4.9 River.

```
570 IF Q = 7 THEN 9600
9600 IF K = 4 THEN 9650
9610 RV = INT (RND (1) *5 + 1): PRINT
      "YOU COME TO A RIVER BANK."
9620 PRINT : K = 4: GOTO 9660
9650 PRINT "YOU ARE AT THE BANK OF A
      RIVER.": PRINT
9660 IF RV = 1 THEN PRINT "IT IS QUITE
      PLEASANT HERE."
9670 IF RV = 3 THEN PRINT "THE SMELL
      OF ANCIENT SEWAGE IS"
9675 IF RV = 3 THEN PRINT "UNPLEASANT,
      BUT BEARABLE."
9680 DG = DG + .25: GOTO 650
```

When K is set to 4, it indicates that the river has already been encountered. On the first encounter, RV is set to a randomly selected value from 1 to 5. RV values of 2, 4, or 5 have no special meaning. However, if RV = 1 the computer will tell you:

"It is quite pleasant here."

On the other hand, if RV = 3 you learn that:

"The smell of ancient sewage is unpleasant, but bearable."

The character's current health rating (DG) benefits slightly by being on a river bank:

```
9680 DG = DG + .25: GOTO 650
```

Special commands for the river will be added later.

MOUNTAIN/RAVINE

The next two obstacles, both dealt with in Listing 4.10, are the mountain (Q = 8) and the ravine (Q = 9).

Listing 4.10 Mountain/Ravine.

```
580 IF Q=8 THEN PRINT "YOU ARE AT THE
      FOOT OF A TALL, CRAGGY MOUNTAIN"
590 IF Q=9 THEN GOSUB 11900
11899 REM * RAVINE *
11900 L3 = L1:L4 = L2: PRINT "YOU JUST
      FELL INTO A DEEP RAVINE!"
11905 DG = DG - INT (RND (1) * DG + 1)
11910 FOR X = 1 TO 4:Y = INT (RND (1)
      * 11 + 1): IF T(Y) = 1 THEN
11980
11920 IF J(Y) = 1 THEN 11990
11930 IF S(Y) > 0 THEN 12000
11940 NEXT : PRINT : GOSUB 10000
11950 PRINT "IT TAKES QUITE A BIT OF
      EFFORT, BUT YOU MANAGE TO CRAWL
      OUT TO ";
11960 PRINT "THE SOUTH.": PRINT :L1 =
      L1 + 1: IF L1 > 10 THEN L1 = 1
```

```

11970 Q = LC (L1,L2):EX (L1,L2) = Q:
      RETURN
11980 T(Y) = 0:E(Y) = INT (RND (1) *
      10 + 1):F(Y) = INT (RND (1) *
      10 + 1): GOTO 11940
11990 J(Y) = 0:C(Y) = INT (RND (1) *
      10 + 1):F(Y) = INT (RND (1) *
      10 + 1): GOTO 11940
12000 S(Y) = 0:A(Y) = INT (RND (1) *
      10 + 1):B(Y) = INT (RND (1) *
      10 + 1): GOTO 11940

```

While special commands will be added later, for now a simple message will be displayed when a mountain is encountered:

```

580 IF Q=8 THEN PRINT "YOU ARE AT THE
      FOOT OF A TALL, CRAGGY MOUNTAIN"

```

On the other hand, encounter of a ravine calls up a subroutine:

```

590 IF Q=9 THEN GOSUB 11900

```

Because of the nature of the ravine subroutine, no commands will be given to deal directly with this particular obstacle.

If the explorer comes across a ravine, he falls in and damages his current health rating (DG) by a random amount. He may also drop some of the items he is carrying. The items will be relocated randomly in nearby locations. He will always crawl out of the ravine to the south, ending up in a new map location. For this reason there are no special commands for the ravine.

MARSQUAKE

A Marsquake is the next obstacle. This is like an earthquake on Earth. A subroutine is called to create the action of the Marsquake:

```

600 IF Q=10 THEN GOSUB 12050

```

The complete subroutine is in Listing 4.11. No commands will affect the Marsquake.

Listing 4.11 Marsquake.

```

600 IF Q = 10 THEN GOSUB 12050
12049 REM * MARSQUAKE *
12050 PRINT "THE GROUND BEGINS TO
      RUMBLE BENEATH YOUR FEET!":PRINT
12060 L1 = INT (RND (1)*5+1) + L1-3:
      IF L1 > 10 THEN L1=1
12070 IF L1 < 1 THEN L1=10
12080 L3 = L1-1: IF L3 < 1 THEN
      L3 = 10
12090 L2 = L2 + INT (RND (1)*5+1) - 3:
      IF L2 < 1 THEN L2 = 10
12100 IF L2 > 10 THEN L2 = 1
12110 L4 = L2: FOR X = 1 TO 40
12120 Y = INT (RND (1) * 10 + 1):Z =
      INT (RND (1) * 10 + 1):ZZ = LC
      (Y,Z): IF ZZ = 20 THEN 12140
12130 IF ZZ > 0 THEN ZZ = - ZZ
12140 LC(Y,Z) = ZZ: NEXT
12150 PRINT " ",:* WHEW! *": PRINT "
      THE MARSQUAKE IS OVER NOW!"
12160 LC(L1,L2) = 0:Q = 0:EX(L1,L2) =
      0: RETURN

```

When the player encounters a Marsquake, a warning message will be displayed:

```

12050 PRINT "THE GROUND BEGINS TO
      RUMBLE BENEATH YOUR FEET!":PRINT

```

The bouncing ground throws the explorer into a new map location. He may be moved from -2 to +2 spaces in either, or both, directions. Of course you must include checks for out of range locations:

```

12060 L1 = INT (RND (1)*5+1) + L1-3:
      IF L1 > 10 THEN L1=1
12070 IF L1 < 1 THEN L1=10
12080 L3 = L1-1: IF L3 < 1 THEN
      L3 = 10

```

```

12090 L2 = L2 + INT (RND (1)*5+1) - 3:
      IF L2 < 1 THEN L2 = 10
12100 IF L2 > 10 THEN L2 = 1
12110 L4 = L2: FOR X = 1 TO 40

```

Next up to 40 map locations are randomly selected. If the location value is 20 (rocket ship), 0 (no occupant), or negative (dead monster) that location will be ignored. For other positive values, the polarity is reversed. That is, the values are made negative, killing the monster, or removing the geographic obstacles.

Note that the changes are made only on the main map array (LC(x,y)). The explored map array (EX(x,y)) is not automatically updated, so the explorer can trust his map just so far. It may contain errors after a Marsquake or certain other events.

If you prefer to make the game easier by automatically updating the explored map array (EX(x,y)), you can add this line:

```

12135 IF EX(Y,Z) > 0 THEN EX(Y,Z) = ZZ

```

STORM

The next obstacle (Q=11) is a Martian storm. The storm is covered completely in a subroutine:

```

610 IF Q=11 THEN GOSUB 12200

```

No special commands will be offered, and storms will not be displayed on the explored map.

The storm subroutine begins by displaying an appropriate message, then the current location is cleared so that each storm will occur only once:

```

12200 PRINT "YOU ARE CAUGHT IN A
      WEIRD MARTIAN STORM!":PRINT
12210 LC(L1,L2) = 0:Q = 0:EX(L1,L2) = 0

```

The character's current health rating is then reduced by up to one fourth of its current value:

```

12220 DG = DG - INT (RND (1) * DG/4+1)

```


As in the Marsquake subroutine, up to 40 percent of the map location occupants are altered by the Martian storm. However, instead of just negating positive values, this subroutine makes negative values positive. Whatever the sign of the randomly selected location may be, its opposite replaces it. This means, for example, that dead monsters may be brought back to life.

Of course, the location containing a value of 20, the rocket ship's location, is protected:

```
12235 IF LC(Y,Z) = 20 THEN 12250
```

By simply negating the current value, we can reverse its sign. For instance $-(-7) = +7$, or $-(+4) = -4$:

```
12240 LC(Y,Z) = - LC(Y,Z)
```

Once again, the changes brought about by the Martian storm are not indicated in the explored map array (EX(x,y)). If you want automatic map up-dating, add this line:

```
12245 EX(Y,Z) = -EX(Y,Z)
```

Notice that there is no need to check for unoccupied locations for bypassing. An unoccupied location is represented by a value of 0, and negating 0 has no effect ($-0 = 0$).

Listing 4.12 is the complete subroutine for the Martian storm.

Listing 4.12 Storm.

```
610 IF Q = 11 THEN GOSUB 12200
12199 REM * STORM *
12200 PRINT "YOU ARE CAUGHT IN A
      WEIRD MARTIAN STORM!":PRINT
12210 LC(L1,L2) = 0:Q = 0:EX(L1,L2) = 0
12220 DG = DG - INT (RND (1) * DG/4+1)
12230 FOR X = 1 TO 40:Y = INT (RND (1) *
      10 + 1):Z = INT (RND (1)* 10 + 1)
12235 IF LC(Y,Z) = 20 THEN 12250
12240 LC(Y,Z) = - LC(Y,Z)
```

```
12250 NEXT : GOSUB 10000
12260 PRINT "THE WEATHER SEEMS TO BE
        CLEARING UP NOW.": PRINT
12270 RETURN
```

FUNNY-COLORED SKY

When the player encounters the final obstacle ($Q=12$), the sky changes color for awhile:

```
620 IF Q=12 THEN GOSUB 12300
12300 PRINT "ODD... THE SKY TURNS A
        FUNNY COLOR FOR A FEW MINUTES..."
12310 PRINT:GOSUB 10000
```

(Subroutine 10000 is the time delay subroutine introduced in Chapter 3.)

Listing 4.13 shows the complete funny-colored sky subroutine. Figure 4.4 outlines the flow-chart.

Listing 4.13 Funny-Colored Sky.

```
620 IF Q = 12 THEN GOSUB 12300
12299 REM * SKY *
12300 PRINT "ODD... THE SKY TURNS A
        FUNNY COLOR FOR A FEW MINUTES..."
12310 PRINT:GOSUB 10000
12320 FOR X = 1 TO 25:Y = INT
        (RND (1) * 10+1): Z=INT (RND
        (1) * 10+1)
12330 ZZ = LC(Y,Z): IF ZZ = 20
        THEN 12360
12335 IF ZZ > 12 THEN ZZ=-1
12340 LC(Y,Z) = ZZ + 1
12360 NEXT:PRINT "WELL, EVERYTHING
        SEEMS TO BE BACK TO NORMAL NOW.";
12370 GOSUB 10000: PRINT "I GUESS...":
        PRINT
12380 RETURN
```

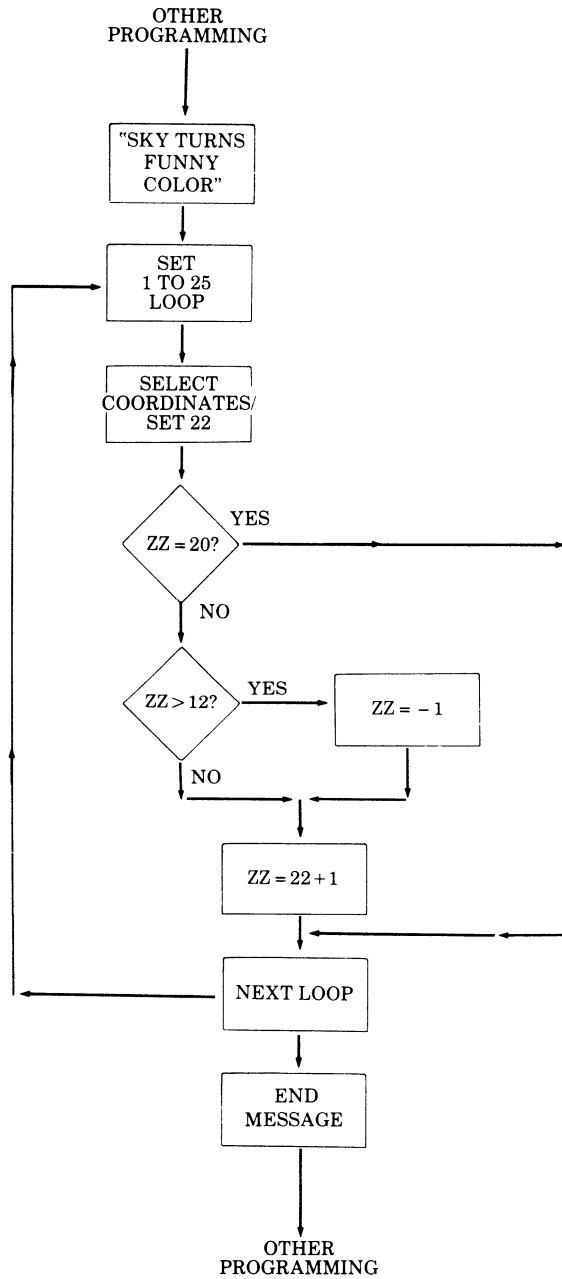


Figure 4.4 Flow-chart for Funny-Colored Sky Routine.

When the sky above Mars changes color, some very strange events take place. The obstacles in up to 25 map locations change into different obstacles. This is done by adding 1 to each of the selected obstacle values:

```

12320 FOR X = 1 TO 25:Y = INT
      (RND (1) * 10+1): Z=INT (RND
      (1) * 10+1)
12330 ZZ = LC(Y,Z): IF ZZ = 20
      THEN 12360
12340 LC(Y,Z) = ZZ + 1
12360 NEXT:PRINT "WELL, EVERYTHING
      SEEMS TO BE BACK TO NORMAL NOW. ";

```

Once again, the special value 20 is protected so the rocket ship will not change.

Table 4.1 shows some of the changes which may be made. The changes are not reflected in the explored map array (EX(x,y)).

Table 4.1 Sample Effects of a Funny-Colored Sky.

Old Value		New Value	
0	blank	1	Squeanly Serpent
1	Squeanly Serpent	2	Ghost
5	Grimph	6	Purofolee
9	Ravine	10	Marsquake
12	funny-colored sky	13	blank
-7	blank	-6	dead Purofolee
-1	dead Squeanly Serpent	0	blank

One additional protection line will avoid problems. For example, a specific location starts out with a value of 12. If this location is selected when the player encounters a funny-colored sky, this value will be raised by 1 to 13. This is an undefined value and will behave no differently from a blank space. There is no problem so far.

But suppose, in the course of a long game, that the same space is selected by funny-colored sky subroutines 7 more times. This would certainly not be impossible. The location would now have a value of 20, the value reserved for the rocket ship. Two rocket ships would now be displayed on the player's map. To prevent this, we can add this line:

```
12335 IF ZZ > 12 THEN ZZ = -1
```

Why is the value set to -1, rather than 0? This is because when 1 is added to the value, the result will be 0, or a blank space.

You might prefer to let phantom rocket ships appear on the map. Since the active location is determined by the values of R1 and R2, only the original rocket ship location will be functional. You can't BLAST OFF, for example, unless you are aboard the real ship, and the game could get complicated if you fail to memorize your ship's original location.

DEAD MONSTERS

Basic routines for each of the 12 obstacles are now added. Before adding commands to deal with some of these obstacles, add a simple subroutine to display dead monsters. As mentioned earlier, a dead monster is indicated by a negative number; for instance, -3 is a dead Brinchley Beast. Simply set up some IF...THEN... tests to display the appropriate message, as shown in Listing 4.14.

Listing 4.14 Dead Monsters.

```
500 Q = LC(L1,L2): PRINT : PRINT
    "YOUR CURRENT COORDINATES ARE ";
10849 REM * DEAD MONSTER *
10850 PRINT "THERE IS A DEAD ";
10860 IF Q = - 1 THEN PRINT "SQEANLEY
    SERPENT";
10870 IF Q = - 2 THEN PRINT "GHOST
    OF AN ANCIENT MARTIAN HERE";
10880 IF Q = - 3 THEN PRINT
    "BRINCHLEY BEAST";
10890 IF Q = - 4 THEN PRINT "KUFU";
10900 IF Q = - 5 THEN PRINT "GRIMPH";
10910 IF Q = - 6 THEN PRINT
    "PUROFOLEE";
10920 PRINT " HERE.":X = INT (RND
    (1) * 10 + 1)
10925 IF (X > 7) OR (Q = -3) OR (Q =
    -5) THEN PRINT "THE SMELL IS
    HORRENDOUS!"
10930 RETURN
```

Except for the PRINT "here." statement, line 10920 is not essential. It simply prints out an extra message; i.e., "The smell is horrendous," under certain conditions. This message will always be displayed for dead Brinchley Beasts ($Q = -3$) and dead Grimphs ($Q = -5$). There is a 30 percent chance that it will be printed for other dead monsters.

MOVING PAST MONSTERS

An obstacle by definition blocks one's path. The monsters should not stand passively by and let the explorer freely use the regular N, S, E, and W move commands. They should block as indicated in Listing 4.15.

Listing 4.15 Moving Past Monsters

```
405 GOSUB 12600
715 IF (Q > 2) AND (Q < 7) THEN 1000
999 REM * MONSTER BLOCK MOVE *
1000 IF Q$ = "N" OR Q$ = "S" OR Q$ =
    "W" OR Q$ = "E" THEN 1010
1005 GOTO 760
1010 IF Q$ = "N" AND MM=1 THEN 5100
1015 IF Q$ = "S" AND MM=2 THEN 5120
1020 IF Q$ = "E" AND MM=3 THEN 5140
1025 IF Q$ = "W" AND MM=4 THEN 5160
1030 IF Q=5 THEN 9100
1040 IF Q=3 THEN PRINT "THE BRINCHLEY
    BEAST BLOCKS YOUR PATH."
1050 IF Q=4 THEN PRINT "THE KUFU WILL
    NOT LET YOU GO THAT WAY.":DG
    = DG-2
1060 IF Q=6 THEN PRINT "THE DURN
    PUROFOLEE IS IN YOUR WAY!"
1070 GOTO 650
9100 PRINT "NONE TOO PLEASED WITH
    YOUR ATTITUDE, ";
12600 IF Q = 3 THEN MM = INT (RND
    (1) * 4 + 1)
12610 IF Q = 4 THEN MM = INT (RND
    (1) * 6 + 1)
```

```

12620 IF Q = 5 THEN MM = INT (RND
      (1) * 10 + 1)
12630 IF Q = 6 THEN MM = INT (RND
      (1) * 5 + 1)
12640 RETURN

```

First off, whenever the player enters a new location, a special subroutine is called:

```

405 GOSUB 12600
12600 IF Q=3 THEN MM=INT (RND (1)*4+1)
12610 IF Q=4 THEN MM=INT (RND (1)*6+1)
12620 IF Q=5 THEN MM=INT (RND (1)*10+1)
12630 IF Q=6 THEN MM=INT (RND (1)*5+1)
12640 RETURN

```

A random value is assigned to the variable MM. The range of possible values is determined by which of the four monsters (Brinchley Beast-3, Kufu-4, Grimp-5, or Purofolee-6) is present. The program uses the variable MM only when Q equals 3, 4, 5, or 6. For other obstacles, it is irrelevant, and this subroutine simply wastes a few microseconds.

Next, you need to add a line in the main command sequence to check for the presence of potentially path blocking monsters. Q will have a value of 3, 4, 5, or 6:

```

715 IF (Q > 2) AND (Q < 7) THEN 1000

```

Next check Q\$ since the blocking routine is only relevant for directional move commands (N, S, E, or W). If the command is not one of the four move commands, the program reverts back to the main command sequence:

```

1000 IF Q$= "N" OR Q$= "S" OR Q$= "W"
      OR Q$= "E" THEN 1010
1005 GOTO 760

```

Now compare the directional command with the value of MM. If MM equals 1, the monster will let the explorer move to the north. If MM equals 2, it will let the explorer move south. For an MM value of 3, it will let him move east. If MM has a value of 4, the monster

will let the player move to the west. If MM has a value greater than 4, the monster will not let the player move at all.

The Brinchley Beast is the most agreeable of the monsters, since it will always allow the player to move in one of the four move directions ($MM = \text{RND}(4)$). The Grimph, on the other hand, is the most troublesome, since 60 percent of the time it won't let the player move in any direction ($MM = \text{RND}(10)$).

If the command and MM value agree, an ordinary move is made:

```
1010 IF Q$= "N" AND MM=1 THEN 5100
1015 IF Q$= "S" AND MM=2 THEN 5120
1020 IF Q$= "E" AND MM=3 THEN 5140
1025 IF Q$= "W" AND MM=4 THEN 5160
```

If the player's attempted move is blocked, an appropriate message is displayed for each type of monster:

```
1040 IF Q=3 THEN PRINT "THE
      BRINCHLEY BEAST BLOCKS YOUR
      PATH. "
1050 IF Q=4 THEN PRINT "THE KUFU
      WILL NOT LET YOU GO THAT WAY. ":DG
      = DG - 2
1060 IF Q=6 THEN PRINT "THE DURN
      PUROFOLEE IS IN YOUR WAY!"
```

Note that if the monster is a Kufu, the current health rating (DG) will be decreased by 2.

The Grimph, the nastiest monster, expresses its displeasure at the explorer's attempted move by attacking:

```
1030 IF Q=5 THEN 9100
9100 PRINT "NONE TOO PLEASED WITH YOUR
      ATTITUDE, ";
*9110 PRINT "THE GRIMPH BREAKS YOUR ";
```

*The Grimph attack routine was in Listing 4.7.

As the program is written here, if the player enters a LOOK command, a new value of MM will be selected.

MOVING PAST RIVERS AND MOUNTAINS

Rivers (Q=7) and mountains (Q=8) will also affect the directional move commands. If a player encounters one of these obstacles, he may only move back the way he came as defined by L3 and L4 (previous location coordinates). Listing 4.16 shows the programming for this routine.

Listing 4.16 Moving Past Rivers/Mountains

```

717 IF (Q = 7 OR Q = 8) AND (Q$ = "N"
    OR Q$ = "S" OR Q$ = "E" OR Q$ =
    "W") THEN 9700
9700 X = L1:Y = L2: IF Q$ = "N" THEN
    X = L1 - 1
9710 IF X < 1 THEN X = 10
9720 IF Q$ = "S" THEN X = L1 + 1
9730 IF X > 10 THEN X = 1
9740 IF Q$ = "E" THEN Y = L2 + 1
9750 IF Y > 10 THEN Y = 1
9760 IF Q$ = "W" THEN Y = L2 - 1
9770 IF Y < 1 THEN Y = 10
9780 IF X = L3 AND Y = L4 THEN 9810
9790 IF Q = 7 THEN PRINT "THE RIVER";
    GOTO 9800
9795 PRINT "THE MOUNTAIN ";
9800 PRINT "IS IN YOUR WAY.":DG=DG
    - 3: GOTO 650
9810 L3 = L1:L4 = L2:L1 = X:L2 = Y:K =
    0: GOTO 400

```

EXPANDING THE CRY COMMAND

The last chapter included the command CRY, which resulted only in display of a sarcastic message. There may be other results from this action when certain monsters are encountered. Listing 4.17 shows the lines required to expand the CRY command.

Listing 4.17 Expanding the CRY Command

```

790 IF QX$ = "CRY" THEN 5180
5179 REM * CRY *
5180 IF Q = 4 THEN 5220

```

```
5190 IF Q = 6 THEN 5240
5200 PRINT "WHY? ARE YOU UPSET FOR
      SOME REASON?"
5210 GOTO 500
5220 PRINT "KUFUS AREN'T NOTED FOR
      BEING SOFTHEARTED."
5230 GOTO 500
5240 X = INT (RND (1) * 10 + 1): IF
      X > 5 THEN 5270
5260 PRINT "ANNOYED BY THE NOISE YOU'RE
      MAKING, THE"
5265 PRINT "PUROFOLEE KNOCKS YOU
      DOWN AND JUMPS ON YOUR STOMACH 17
      TIMES!"
5266 DG = DG - 17: GOTO 500
5270 PRINT "FRIGHTENED BY THE NOISE
      YOU'RE MAKING, THE PUROFOLEE HOPS
      OFF."
5280 X = L1 + INT (RND (1) * 5 + 1)
      - 3: Y = L2 + INT (RND (1) * 5
      + 1) - 3: IF X = L1 AND Y = L2 THEN
5280
```

This command CRY will have special results when facing a Kufu (Q = 4) and a Purofolee (Q = 6).

If the player cries in front of a Kufu, a special message is displayed:

```
5180 IF Q=4 THEN 5220
5220 PRINT "KUFUS AREN'T NOTED FOR
      BEING SOFT-HEARTED."
5230 GOTO 500
```

CRYing in front of a Purofolee badly disturbs this creature, and there is a 50 percent chance that it will attack:

```
5190 IF Q=6 THEN 5240
5240 X = INT(RND (1)*10+1):IF X > 5
      THEN 5270
5260 PRINT "ANNOYED BY THE NOISE YOU'RE
      MAKING, THE"
```

```

5265 PRINT "PUROFOLEE KNOCKS YOU
        DOWN AND JUMPS ON YOUR STOMACH
        17 TIMES!"
5266 DG = DG - 17: GOTO 500

```

The other 50 percent of the time, CRYing will frighten the Purofolee off to be randomly relocated in a near-by location. Line 5280 includes a check to prevent the original location from being reselected.

Similarly, line 5285 prevents the rocket ship's location from being chosen as the Purofolee's new location:

```

5270 PRINT "FRIGHTENED BY THE NOISE
        YOU'RE MAKING, THE PUROFOLEE
        HOPS OFF."
5280 X = L1 + INT (RND (1)*5+1)-3:
        Y=L2+INT (RND (1)*5+1)-3: IF X =
        L1 AND Y = L2 THEN 5280
5285 IF X = R1 AND Y = R2 THEN 5280
5290 LC(L1,L2)=0: EX(L1,L2)=0: Q=0:
        LC(X,Y)=6: GOTO 500

```

Note that any previous occupant of the selected location will be erased and replaced by the Purofolee.

EXPANDING THE EAT COMMAND

Listing 4.18 shows a number of additions to the EAT command which was originally programmed in the last chapter.

Listing 4.18 Expanding the EAT Command

```

5300 IF Q > 3 AND Q < 7 THEN 5600
5320 IF QY$ = "UFU" THEN 5480
5330 IF QY$ = "LEE" THEN 5540
5480 IF Q = -4 THEN 5500
5490 PRINT "FIRST YOU HAVE TO GO OUT
        AND KILL ONE, ";N$:DG = DG - 1
        : GOTO 500
5500 PRINT "YOU HAVE TO HOLD YOUR
        NOSE TO GET CLOSE ENOUGH,"
5510 DG = DG + INT (RND (1) * 15 +
        1) - 10: IF DG > DX THEN DG = DX

```

```
5520 PRINT "BUT YOU SOMEHOW MANAGE TO
      GET IT DOWN.": LC(L1,L2) = 0
5530 EX(L1,L2) = 0:Q = 0: GOTO 500
5540 IF Q = - 6 THEN 5560
5550 GOTO 5490
5560 PRINT " ",:YUMMY!": PRINT :DG =
      DG + 50
5570 IF DG > DX THEN DG = DX
5580 LC(L1,L2) = 0:EX(L1,L2) = 0:Q =
      0: GOTO 500
5600 IF Q = 4 THEN 5640
5610 IF Q = 5 THEN 5670
5620 IF S(1) = 0 THEN PRINT "THERE IS
      NOTHING HERE TO EAT.":DG = DG
      - 1: GOTO 500
5630 PRINT "WHEN YOU TAKE YOUR FOOD
      OUT, THE DURN"
5635 PRINT "PUROFOLEE SNATCHES
      IT BEFORE YOU CAN TAKE A SINGLE
      BITE!"
5636 PRINT :S(1) = 0:DG = DG
      -1: GOTO 500
5640 PRINT "THIS IS NO TIME TO THINK
      OF YOUR STOMACH, ";N$;:"!"
5650 GOSUB 10000: PRINT "UNLESS YOU
      WANT TO SEE THE INSIDE OF A
      KUFU'S STOMACH!"
5660 DG = DG - 2: GOTO 500
5670 PRINT "THE GRIMPH THINKS THAT'S
      A JIM DANDY IDEA!": PRINT : GOSUB
      10000
5680 PRINT "IT EATS YOU!": PRINT : GOTO
      5070
```

The first step when an EAT command is recognized is to see if one of three monsters—Kufu (Q=4), Grimp (Q=5), or Purofolee (Q=6)—is present. The Brinchley Beast, ghost, and Squeanly Serpent will not respond to the explorer EATing:

```
5300 IF Q > 3 AND Q < 7 THEN 5600
```

If the player enters an EAT command while facing a Kufu, a warning message will be displayed, and his current health rating (DG) will be penalized two points:

```
5600 IF Q=4 THEN 5640
5640 PRINT "THIS IS NO TIME TO THINK
      OF YOUR STOMACH, ";N$;"!"
5650 GOSUB 10000:PRINT "UNLESS YOU
      WANT TO SEE THE INSIDE OF A
      KUFU'S STOMACH!"
5660 DG=DG-2: GOTO 500
```

Subroutine 10000 is the time delay loop.

To EAT in front of a Grimp is a particularly bad idea. It thinks you are making a suggestion and eats you:

```
5610 IF Q=5 THEN 5670
5670 PRINT "THE GRIMPH THINKS THAT'S
      A JIM DANDY IDEA!":PRINT:
      GOSUB 10000
5680 PRINT "IT EATS YOU!":PRINT:GOTO
      5070
```

The character dead/end-of-game routine is at 5070. This was programmed at an earlier stage.

If you try to EAT in front of a Purofolee when you don't have any food, you will be reminded that there is nothing available to eat:

```
5620 IF S(1) = 0 THEN PRINT "THERE IS
      NOTHING HERE TO EAT.":DG = DG - 1:
      GOTO 500
```

However, if the explorer is carrying the food, he won't be for long. Purofolees, natural thieves that they are, will steal the food before the character can eat it:

```
5630 PRINT "WHEN YOU TAKE YOUR FOOD OUT,
      THE DURN"
5635 PRINT "PUROFOLEE SNATCHES IT
      BEFORE YOU CAN TAKE A SINGLE BITE!":
5636 PRINT :S(1)=0: DG=DG-1: GOTO 500
```

So far, the food from the supplies is the only thing the character can eat. If you add a few more lines, the player can make a feast of certain dead monsters. Only Kufus and Purofolees will be recognized as edible.

Start with the Kufu:

```
5320 IF QY$= "UFU" THEN 5480
```

Naturally, a dead Kufu must be present in order for the explorer to eat it. A dead Kufu is represented by a Q value of -4. Check for it like this:

```
5480 IF Q = -4 THEN 5500
5490 PRINT "FIRST YOU HAVE TO GO OUT AND
      KILL ONE,";N$: DG=DG-1: GOTO 500
```

A Kufu is not the tastiest thing on Mars. Lines 5500 and 5520 inform you that:

You have to hold your nose to get close enough, but you somehow manage to get it down.

Lines 5520 and 5530 also clear the map array location. After all, once the Kufu is eaten, its carcass shouldn't still be lying there.

EATING a dead Kufu is an unreliable source of nourishment. It may increase the current health rating by up to 5, or it may decrease it up to 10 points:

```
5510 DG = DG + INT (RND (1) * 15+1) - 10:
      IF DG > DX THEN DG = DX
```

A much better meal may be made from a dead Purofolee:

```
5330 IF QY$ = "LEE" THEN 5540
```

Once again, a dead Purofolee ($Q = -6$) must be present for the explorer to eat it:

```
5540 IF Q = - 6 THEN 5560
5550 GOTO 5490
```

If a dead Purofolee is available, EATING it proves to be both delicious and nutritious, adding 50 points to the current health rating (DG):

```
5560 PRINT " ", "YUMMY!": PRINT:
      DG = DG + 50
```

Of course the current health rating (DG) is not permitted to exceed the maximum health rating (DX):

```
5570 IF DG > DX THEN DG = DX
```

Once the Purofolee is eaten, the map location is cleared to get rid of the body:

```
5580 LC(L1,L2) = 0: EX(L1,L2) =
      0: Q = 0: GOTO 500
```

EXPANDING THE DRINK COMMAND

It is not reasonable for an explorer to DRINK from his bottle of water when he is at a river ($Q = 7$). Listing 4.19 includes programming that lets the explorer drink directly from the river when appropriate.

Listing 4.19 Expanding the DRINK Command

```
5700 IF Q = 7 THEN 5810
5810 PRINT " ", "SLURP!!!":
      PRINT: GOSUB 10000
5820 X = INT(RND (1) * 10 + 1):
      IF X > 7 THEN 5860
5830 IF X < 4 THEN 5870
5840 DG = DG + DX * .15: IF
      DG > DX THEN DG = DX
5850 GOTO 500
5860 PRINT "IT SURE TASTES GOOD!": DG
      = DG + DX * .1: GOTO 5840
5870 PRINT "THIS WATER MAKES YOU
      QUITE ILL.": DG = DG * .75
```

```
5880 PRINT "BUT YOU DON'T DIE";:
      GOSUB 10000
5890 PRINT "--- AT LEAST, NOT
      JUST YET.":GOTO 500
```

Ordinarily, drinking from a river improves the character's current health rating by 15 percent of the maximum value, provided that the current rating (DG) does not exceed the maximum rating (DX):

```
5700 IF Q = 7 THEN 5810
5810 PRINT " ", "SLURP!!!":
      PRINT: GOSUB 10000
5840 DG = DG + DX * .15: IF
      DG > DX THEN DG = DX
5850 GOTO 500
```

To add a little variety, a random number from 1 to 10 is selected by the computer. If this number is greater than 7 (30 percent chance), the water is particularly good. Besides being tasty, it adds an extra 10 percent of the maximum health rating (DX) to the current health rating (DG) for a total gain of up to 25 percent of DX:

```
5860 PRINT "IT SURE TASTES GOOD!":DG
      = DG + DX * .1: GOTO 5840
```

Notice that the player does not have to do the $DG > DX$ test here since it will be performed in line 5840 anyway.

If the randomly selected value (X) is less than 4 (again, a 30 percent chance), the water in the river turns out to be polluted and drops the current health rating (DG) by 25 percent, leaving it at 75 percent of its original level:

```
5830 IF X < 4 THEN 5870
5870 PRINT "THIS WATER MAKES YOU
      QUITE ILL.":DG = DG * .75
5880 PRINT "BUT YOU DON'T DIE";:
      GOSUB 10000
5890 PRINT "--- AT LEAST, NOT
      JUST YET.":GOTO 500
```


There is a 40 percent chance ($X = 4, 5, 6$, or 7) that the river will contain just ordinary water.

ADDING THE FILL COMMAND

As mentioned earlier, the player can refill his water bottle at a river. To do so he enters the FILL command. Listing 4.20 contains the programming for this new command that will only be recognized when the explorer is at a river ($Q = 7$).

Listing 4.20 Fill.

```

820 IF Q = 7 AND QX$ = "FIL"
    THEN 5900
5899 REM * FILL *
5900 X = LEN (Q$): IF X > 6
    THEN 5920
5910 PRINT "FILL WHAT?": DG = DG - .5:
    GOTO 500
5920 IF QY$ = "TLE" THEN 5990
5930 IF QY$ = "URN" THEN 6040
5940 IF QY$ = "BOX" THEN 6060
5950 IF QY$ = "UIT" THEN PRINT
    YOU DROWN!": GOTO 5070
5960 IF QY$ = "OWL" THEN 6080
5970 IF QY$ = "CUP" THEN 6090
5980 PRINT "THAT WON'T HOLD WATER, "; N$:
    DG = DG - 1: GOTO 500
5990 IF S(2) = 2 THEN 6030
6000 IF S(2) = 1 THEN 6020
6010 PRINT "YOU DON'T HAVE IT, "; N$; "!":
    DG = DG - 1.25: GOTO 500
6020 PRINT "IT'S ALREADY FULL": DG=
    DG - 1: GOTO 500
6030 PRINT "OK. THE BOTTLE IS NOW
    FULL.": S(2) = 1: GOTO 500
6040 IF J(8) = 0 THEN 6010
6045 PRINT "THE URN";
6050 PRINT "LEAKS.": DG = DG - .75:
    GOTO 500
6060 IF T(9) = 0 THEN 6010
6070 PRINT "THE BOX DISSOLVES.":
    T(9) = 0: GOTO 500

```

```
6080 IF T(1) = 0 THEN 6010
6085 PRINT "THE COPPER BOWL";:
      GOTO 6050
6090 IF T(5) = 0 THEN 6010
6095 PRINT "THE SILVER CUP";:
      GOTO 6050
```

The player must specify an object to be filled with a command that is at least 6 characters long or no object will be specified and an error message is displayed:

```
820 IF Q = 7 AND QX$ = "FIL"
    THEN 5900
5900 X = LEN (Q$):IF X > 6
    THEN 5920
5910 PRINT "FILL WHAT?":DG = DG - .5:
    GOTO 500
```

The string variable QY\$ has already been set to equal the last three characters of the complete command. Use this substring to determine which object is specified.

The bottle (QY\$ = "TLE") is the logical choice:

```
IF QY$ = "TLE" THEN 5990
```

The computer must make two checks before the bottle can be filled. First, the explorer must be carrying it; and second, it must be empty. If the bottle is empty S(2) will hold a value of 2. You only need to change this to 1 (to indicate a full bottle) and display an appropriate message:

```
5990 IF S(2) = 2 THEN 6030
6030 PRINT "OK. THE BOTTLE IS NOW
      FULL. ": S(2) = 1: GOTO 500
```

If the bottle is already full (S(2)=1), the player cannot fill it again:

```
6000 IF S(2) = 1 THEN 6020
6020 PRINT "IT'S ALREADY FULL": DG
      = DG-1: GOTO 500
```

If S(2) fails both of these tests (has a value that is neither 1 nor 2), it is assumed to equal 0, meaning the character does not have the bottle. You can't fill a bottle you don't have:

```
6010 PRINT "YOU DON'T HAVE IT, ";N$;"!":
      DG = DG - 1.25: GOTO 500
```

In preliminary testing of this game, players often tried to FILL some of the other objects when the bottle was not available. These objects included the urn, the humming box, the copper bowl, and the silver cup. One player even tried to FILL his spacesuit. So for a nice touch, the program recognizes these objects and responds accordingly. Trying to FILL these objects will produce special error messages and, in some cases, problems for the player.

A universal error message should also be included, in case a player comes up with a possibility you didn't think of while programming the game:

```
5980 PRINT "THAT WON'T HOLD WATER, ";N$:
      DG = DG - 1: GOTO 500
```

For the urn, copper bowl, and silver cup, just print a message that they leak and subtract 75 percent from the current health rating (DG). You must also include checks to make sure the character is actually carrying the specified object:

```
5930 IF QY$ = "URN" THEN 6040
5960 IF QY$ = "OWL" THEN 6080
5970 IF QY$ = "CUP" THEN 6090
6040 IF J(8) = 0 THEN 6010
6045 PRINT "THE URN";
6050 PRINT "LEAKS.": DG = DG - .75:
      GOTO 500
6080 IF T(1) = 0 THEN 6010
6085 PRINT "THE COPPER BOWL";:
      GOTO 6050
6090 IF T(5) = 0 THEN 6010
6095 PRINT "THE SILVER CUP";:
      GOTO 6050
```

The humming box (T(9)) is a special object. An attempt to fill it with water will be penalized, and the box will dissolve to be permanently lost:

```
5940 IF QY$ = "BOX" THEN 6060
6060 IF T(9) = 0 THEN 6010
6070 PRINT "THE BOX DISSOLVES. ":
      T(9) = 0: GOTO 500
```

As for the player who tried to fill his spacesuit with water, a drastic penalty awaits:

```
5950 IF QY$ = "UIT" THEN PRINT
      "YOU DROWN!": GOTO 5070
```

The attempt is fatal.

ADDING INFLATE COMMAND

Since one of the supplies is an inflatable raft (S(7)), an INFLATE command is appropriate. Listing 4.21 shows the programming for this new command.

Listing 4.21 Inflate.

```
830 IF S(7)=1 AND QX$= "INF"
      THEN 6100
6099 REM * INFLATE RAFT *
6100 PRINT " ", "PHFFFTT!": PRINT
6110 GOSUB 10000: IF Q = 7 THEN 6130
6120 PRINT "THE RAFT SPRINGS A LEAK!":
      PRINT :S(7) = 0:GOTO 500
6130 DG = DG - (INT (RND (1) * 50 + 1)
      / 10)
6140 PRINT "YOU CANNOT CONTROL THE
      RAFT ON THESE RAGING CURRENTS"
6150 PRINT : GOSUB 10000
6160 PRINT "YOU MAKE IT TO A SHORE,
      BUT YOU LOSE YOUR RAFT!"
```

```

6170 X = L1 - 2 + INT (RND (1) * 3 +
      1):Y = L2 - 2 + INT (RND (1) *
      3 + 1): IF X = L1 AND Y = L2 THEN
      6170
6172 IF X < 1 THEN X = 10
6174 IF X > 10 THEN X = 1
6176 IF Y < 1 THEN Y = 10
6178 IF Y > 10 THEN Y = 1
6180 S(7) = 0:L3 = L1:L4 = L2:L1 = X:
      L2 = Y: INPUT "PLEASE PRESS
      'RETURN'      ";Q$
6190 GOTO 400

```

The INFLATE command will only be recognized by the program when the explorer is carrying the raft:

```

830 IF S(7)=1 AND QX$= "INF"
      THEN 6100

```

Of course, it only makes sense to inflate your raft when you are at a river (Q = 7), so something should happen if a player enters the command at some other time. To penalize the player for this illogical command, have the raft spring a leak and become useless:

```

6110 GOSUB 10000: IF Q=7 THEN
      6130
6120 PRINT "THE RAFT SPRINGS A
      LEAK!":PRINT :S(7) = 0:
      GOTO 500

```

The player can INFLATE his raft to cross a river, but you don't want things to be too easy. The river crossing will decrease the current health rating (DG), the raft will be lost, and the player will have no way to steer. His new location will be randomly determined (line 6170).

ADDING CLIMB COMMAND

When facing a mountain (Q = 8), the player can only move back the way he came (as determined by L3 and L4). Sometimes he may have to get past a mountain in another direction. To do this, add a

new command—CLIMB. This command will only be recognized when there is a mountain present:

```
850 IF Q = 8 AND QX$ = "CLI"  
    THEN 6200
```

When the CLIMB command is entered, the player is asked which way he wants to go. He must enter one of the four directional commands (N, S, E, or W):

```
6200 INPUT "DIRECTION?";Q$: Q$ =  
    LEFT$ (Q$, 1)  
6210 IF (Q$ = "N") OR (Q$ = "S") OR  
    (Q$ = "E") OR (Q$ = "W")  
    THEN 6230  
6220 GOTO 6200
```

You might want to include a way to let the player change his mind. Adding the following line allows the player to enter X to cancel the CLIMB command:

```
6205 IF Q$ = "X" THEN 700
```

After the player enters the desired direction, the computer checks to see if the coil of rope (S(6)) is being carried:

```
6230 IF S(6) = 1 THEN 6270
```

For now we will assume that the character does not have the rope. He selects a random number from 1 to 10:

```
6240 X = INT (RND (1) * 10+1)
```

Then the computer checks to see if the explorer is carrying the slimy thing (J(12)):

```
6250 IF J(12) = 1 THEN 6320
```

If the player does not have the slimy thing, there is a 30 percent chance that he will successfully climb the mountain (program

control jumps back to line 720 where the main control sequence makes a normal directional move):

```
6260 IF X > 7 THEN 720
6265 GOTO 6360
```

There is a 70 percent chance that the explorer will fall off the side of the mountain and reduce his current health rating (DG) by a randomly determined amount:

```
6360 PRINT "YOU FALL!"
6370 DG = DG - INT (RND (1) *
      DG + 1) + 1
6380 GOTO 500
```

If the explorer is carrying the slimy thing, but not the rope, there is a 33 percent chance of the above catastrophe:

```
6320 Y = INT (RND (1) * 12 + 1):
      IF Y<5 THEN 6260
```

There is a 66 percent chance that the explorer will drop the slimy thing while part way up the mountain:

```
6330 IF S(12) = 0 THEN 6360
6335 PRINT "HALF WAY UP THE
      MOUNTAIN, YOU DROP THE SLIMY
      THING!"
```

There is now about a 57 percent chance that the character will slip on the dropped slimy thing and fall off of the mountain. Otherwise, it reverts back to line 6260 for the regular no rope chance of falling:

```
6340 J(12) = 0: C(12) = L1: D(12)
      = L2: IF Y>8 THEN 6260
6350 PRINT "YOU TRIP ON THE SLIMY
      THING AND FALL!":GOTO 6370
```

If, on the other hand, the character is carrying the rope, he will be asked if he wants to use it in the climb:

```
6270 INPUT "DO YOU USE YOUR  
      ROPE";X$:X$ = LEFT$ (X$,1)  
6280 IF X$ = "Y" THEN 6300  
6290 GOTO 6240
```

If for some reason the player responds NO, everything will proceed as if the rope was not present. If he says YES, the fall factor (X) will be given a random value of 1 to 20, instead of 1 to 10. Ignoring the slimy thing (which functions in the same way with or without the rope), the chances of successfully climbing the mountain are increased from 30 percent to 35 percent.

Listing 4.22 Climb.

```
850 IF Q = 8 AND QX$ = "CLI" THEN  
      6200  
6199 REM * CLIMB MOUNTAIN *  
6200 INPUT "DIRECTION?";Q$: Q$ =  
      LEFT$ (Q$,1)  
6205 IF Q$ = "X" THEN 700  
6210 IF (Q$ = "N") OR (Q$ = "S") OR  
      (Q$ = "E") OR (Q$ = "W")  
      THEN 6230  
6220 GOTO 6200  
6230 IF S(6) = 1 THEN 6270  
6240 X = INT (RND (1) * 10+1)  
6250 IF J(12) = 1 THEN 6320  
6260 IF X > 7 THEN 720  
6265 GOTO 6360  
6270 INPUT "DO YOU USE YOUR  
      ROPE";X$:X$ = LEFT$ (X$,1)  
6280 IF X$ = "Y" THEN 6300  
6290 GOTO 6240  
6300 X = INT (RND (1) * 20 + 1): IF  
      J(12) = 1 THEN 6320  
6310 GOTO 6260  
6320 Y = INT (RND (1) * 12 + 1):  
      Y<5 THEN 6260  
6330 IF S(12) = 0 THEN 6360  
6335 PRINT "HALF WAY UP THE  
      MOUNTAIN, YOU DROP THE SLIMY  
      THING!"
```



```

6340 J(12) = 0: C(12) = L1: D(12)
    = L2: IF Y>8 THEN 6260
6350 PRINT "YOU TRIP ON THE SLIMY
    THING AND FALL!":GOTO 6370
6360 PRINT "YOU FALL!"
6370 DG = DG - INT (RND (1) *
    DG + 1) + 1
6380 GOTO 500

```

OPEN BOX

Listing 4.23 shows the programming to add a command to open the mysteriously humming box (T(9)). The character must be carrying the box for this command to be recognized. No other object may be opened.

Listing 4.23 Open Box.

```

870 IF T(9) = 1 AND Q$ = "OPEN BOX"
    THEN 6400
6399 REM * OPEN BOX *
6400 PRINT: PRINT " ", "HMMMM....":
    PRINT: GOSUB 10000
6410 IF Q = 6 THEN 6470
6420 IF Q = 4 THEN 6500
6430 FOR X = 1 TO 3:Y = INT (RND (1)
    * 12 + 1): IF J(Y) = 1 THEN J(Y)=0
6440 NEXT:IF QX$= "PRA" THEN 500
6450 PRINT "THE BOX SNAPS SHUT!":
    PRINT " ", "CLICK":PRINT
6460 GOTO 500
6470 PRINT:PRINT " ", "*** POOF ***":
    PRINT
6480 GOSUB 10000:Q = 0: LC(L1,L2) = 0:
    EX(L1,L2)=0
6490 PRINT "THE PUROFOLEE VANISHES.":
    PRINT:GOSUB 10000:GOTO 6450
6500 PRINT " ", "*** POOF ***":PRINT:
    GOSUB 10000
6510 LC(L1,L2) = 5:EX(L1,L2) = 5:Q = 5
6520 PRINT "THE KUFU JUST TURNED INTO
    A GRIMPH!"
6530 GOTO 500

```

Ordinarily, when the box is opened, it hums for a few seconds, and then snaps shut. Up to three of the junk items being carried by the explorer may vanish without a trace:

```
870 IF T(9) = 1 AND Q$ = "OPEN BOX"
    THEN 6400
6400 PRINT: PRINT " ", "HMMMM....":
    PRINT: GOSUB 10000
6430 FOR X = 1 TO 3: Y = INT (RND (1)
    * 12 + 1): IF J(Y) = 1 THEN J(Y)=0
6440 NEXT: IF QX$ = "PRA" THEN 500
6450 PRINT "THE BOX SNAPS SHUT!":
    PRINT " ", "CLICK": PRINT
6460 GOTO 500
```

OPENing the box in front of a Purofolee (Q=6) has a unique effect—it causes the Purofolee to vanish into thin air:

```
6410 IF Q = 6 THEN 6470
6470 PRINT: PRINT " ", "*** POOF ***":
    PRINT
6480 GOSUB 10000: Q = 0: LC(L1,L2) = 0:
    EX(L1,L2)=0
6490 PRINT "THE PUROFOLEE VANISHES.":
    PRINT: GOSUB 10000: GOTO 6450
```

OPENing the box when a Kufu is present, also has a special effect—the Kufu is magically transformed into a Grimph (an even nastier monster):

```
6500 PRINT " ", "*** POOF ***": PRINT:
    GOSUB 10000
6510 LC(L1,L2) = 5: EX(L1,L2) = 5: Q = 5
6520 PRINT "THE KUFU JUST TURNED INTO
    A GRIMPH!"
6530 GOTO 500
```

OPENing the box has no special results in front of a Grimph, Brinchley Beast, or other obstacle.

EXPANDING PRAY COMMAND

When we set up the PRAY command in Chapter 3, we left space to add special results of praying when facing monsters.

If you PRAY in front of a Brinchley Beast and are not carrying the statue of the three-armed Martian god, not much will come of the action:

```
6550 IF Q = 3 THEN 6600
6600 IF T(4) = 1 THEN 6620
6610 PRINT "NOTHING MUCH SEEMS TO
      HAPPEN. ":DG = DG + 1:GOTO 500
```

However, if you have the statue ($T(4)=1$), the results are dramatic—the relatively harmless Brinchley Beast vanishes and is replaced by a hungry Kufu:

```
6620 PRINT:PRINT " ", "*** POOF ***":
      PRINT:GOSUB 10000
6630 PRINT "THE BRINCHLEY BEAST TURNS
      INTO A KUFU!"
6640 LC(L1,L2) = 4: EX(L1,L2) = 4: Q =
      4: GOTO 500
```

That three-armed Martian god is apparently not too fond of earthlings.

PRAYing in front a Kufu is also not much of a help. The Martian gods ignore you, and the monster takes the chance to attack:

```
6560 IF Q = 4 THEN 6650
6650 PRINT "KUFUS AREN'T KNOWN FOR
      BEING PIOUS":PRINT:GOSUB 10000
6660 PRINT "IT TAKES ADVANTAGE OF YOUR
      INATTENTION TO ATTACK!":PRINT
```

The Kufu attack sequence (beginning at line 6670) was written earlier in this chapter.

Listing 4.24 Pray.

```
880 IF QX$ = "PRA" THEN 6550
6440 NEXT:IF QX$= "PRA" THEN 500
```

```
6550 IF Q = 3 THEN 6600
6560 IF Q = 4 THEN 6650
6570 IF Q = 5 THEN 6760
6580 IF Q = 6 THEN 6800
6590 DG = DG + 1: PRINT : PRINT "GIMME
      THAT OLD TIME RELIGION!": PRINT:
      GOTO 500
6600 IF T(4) = 1 THEN 6620
6610 PRINT "NOTHING MUCH SEEMS TO
      HAPPEN.":DG = DG + 1:GOTO 500
6620 PRINT:PRINT " ", "*** POOF ***":
      PRINT:GOSUB 10000
6630 PRINT "THE BRINCHLEY BEAST TURNS
      INTO A KUFU!"
6640 LC(L1,L2) = 4: EX(L1,L2) = 4: Q =
      4: GOTO 500
6650 PRINT "KUFUS AREN'T KNOWN FOR
      BEING PIOUS":PRINT:GOSUB 10000
6660 PRINT "IT TAKES ADVANTAGE OF YOUR
      INATTENTION TO ATTACK!":PRINT
6760 PRINT:PRINT "SORRY --- ": PRINT:
      GOSUB 10000
6770 PRINT "ALL MARTIAN DEITIES ARE BUSY
      AT THE MOMENT.":GOSUB 10000
6780 PRINT "PLEASE CALL AGAIN.": PRINT:
      GOSUB 10000
6790 PRINT " ", "HAVE A NICE DAY!!":
      PRINT: PRINT:GOTO 500
6800 IF S(5)=1 THEN 6760
```

PRAYing won't help much when you deal with a Grimph. The Martian gods politely brush you off:

```
6570 IF Q = 5 THEN 6760
6760 PRINT:PRINT "SORRY --- ": PRINT:
      GOSUB 10000
6770 PRINT "ALL MARTIAN DEITIES ARE BUSY
      AT THE MOMENT.":GOSUB 10000
6780 PRINT "PLEASE CALL AGAIN.": PRINT:
      GOSUB 10000
6790 PRINT " ", "HAVE A NICE DAY!!":
      PRINT: PRINT:GOTO 500
```

If your problem is a Purofolee, PRAYing may or may not help. The Martian gods are somewhat pacifistic, and if you are carrying a laser, your prayers will be ignored, and you'll get the same message for PRAYing in front of a Grimph.

However, if you do not have the laser ($S(5) = 0$), the Purofolee will vanish:

```
6580 IF Q = 6 THEN 6800
6800 IF S(5) = 1 THEN 6760
```

We can use the same Purofolee vanishing routine set up for the OPEN BOX command, if we change line 6440 to read:

```
6440 NEXT:IF QX$ = "PRA" THEN 500
```

For all other obstacles, the PRAY command functions in its normal manner, as described back in Chapter 3.

KILL COMMAND

Sooner or later, the explorer will find himself facing a monster that won't let him pass no matter what he does. When this happens, the player has only one choice—KILL it. Listing 4.25 lists the programming for the KILL command. This routine is rather lengthy, but not very complex.

The KILL command is really only appropriate when facing one of the four active monsters—Brinchley Beast, Kufu, Grimph, or Purofolee. After all, it is not appropriate to kill a river.

If you use the KILL command inappropriately, the computer displays a sarcastic message, and subtracts two points from the character's current health rating (DG):

```
7900 IF (Q>2) AND (Q<7) THEN 7930
7910 PRINT "MY, BUT WE'RE IN A HOSTILE
      MOOD TODAY!"
7920 DG = DG-2: GOTO 500
```

When you decide to KILL a monster, you must select a weapon:

```
7930 INPUT "YOUR CHOICE OF WEAPON";Q$
```

Ten potential weapons are recognized by the program. They are KNIFE (S(3)), GUN (S(4)), LASER (S(5)), ROPE (S(6)), METAL PIPE (S(9)), ROCK (J(3)), SHARPENED STICK (J(7)), PETRIFIED WAD OF BUBBLE GUM (J(9)), SLIMY THING (J(12)), and LARGE SWORD (T(10)). Anything else entered causes an error message to be displayed:

```
7990 PRINT "THAT DOESN'T SOUND LIKE A  
      VERY GOOD WEAPON TO ME,";N$  
7995 GOTO 500
```

For simplicity, in the following discussion we will concentrate on the knife (S(3)). The other weapons are similarly programmed.

First, lines 7940 through 7985 are used to recognize the weapon name. A value is assigned to the variable W, depending on which weapon has been selected. For the knife, W is set equal to 1. The program then jumps to line 8000:

```
7940 IF Q$ = "KNIFE" THEN W = 1:  
      GOTO 8000
```

In lines 8000 through 8045, checks are made to make sure the explorer is carrying the specified weapon. If he is, the program jumps to an appropriate line number:

```
8000 IF W = 1 AND S(3) = 1 THEN 8100
```

If the specified item is not being carried, an error message is displayed, and the monster (except for the Purofolee) attacks the explorer. This is done in lines 8050 through 8090 in Listing 4.25.

Listing 4.25 Kill.

```
910 IF QX$ = "KIL" THEN 7900  
7899 REM * KILL *  
7900 IF (Q)2) AND (Q)7) THEN 7930  
7910 PRINT "MY, BUT WE'RE IN A HOSTILE  
      MOOD TODAY!"  
7920 DG = DG-2: GOTO 500
```

```
7930 INPUT "YOUR CHOICE OF WEAPON";Q$
7940 IF Q$ = "KNIFE" THEN W = 1:
      GOTO 8000
7950 IF Q$ = "LASER" THEN W = 3: GOTO
      8000
7955 QY$ = RIGHT$(Q$,4): IF QY$ =
      "ROPE" OR QY$ = "COIL" THEN W = 4
      : GOTO 8000
7960 IF QY$ = "PIPE" THEN W = 5: GOTO
      8000
7965 IF QY$ = "ROCK" THEN W = 6: GOTO
      8000
7970 IF QY$ = "TICK" THEN W = 7: GOTO
      8000
7975 IF QY$ = " WAD" OR QY$ = " GUM"
      THEN W = 8: GOTO 8000
7980 IF QY$ = "HING" THEN W = 9: GOTO
      8000
7985 IF QY$ = "WORD" THEN W = 10: GOTO
      8000
7990 PRINT "THAT DOESN'T SOUND LIKE A
      VERY GOOD WEAPON TO ME,";N$
7995 GOTO 500
8000 IF W = 1 AND S(3) = 1 THEN 8100
8005 IF W = 2 AND S(4) = 1 THEN 8200
8010 IF W = 3 AND S(5) = 1 THEN 8300
8015 IF W = 4 AND S(6) = 1 THEN 8400
8020 IF W = 5 AND S(9) = 1 THEN 8500
8025 IF W = 6 AND J(3) = 1 THEN 8600
8030 IF W = 7 AND J(7) = 1 THEN 8700
8035 IF W = 8 AND J(9) = 1 THEN 8800
8040 IF W = 9 AND J(12) = 1 THEN 8900
8045 IF W = 10 AND T(10) = 1 THEN 9000
8050 PRINT "YOU DON'T HAVE IT!!": PRINT
8060 IF Q = 3 THEN 7780
8070 IF Q = 4 THEN PRINT "THE KUFU IS
      PLEASED BY YOUR ERROR!": GOTO
      6670
8080 IF Q = 5 THEN 9100
8090 DG = DG - 3: GOTO 500
8099 REM * KNIFE *
8100 X = SX + PX + (AX/2): DG = DG-10
```

```
8110 IF Q = 3 THEN 8160
8120 IF Q = 4 THEN 8170
8130 IF Q = 5 THEN 8180
8140 Y = INT (RND (1) * 200+1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
8150 PRINT " ", "GIB-BLECK...": GOTO
      8190
8160 Y = INT (RND (1)*200+1): IF Y>X
      THEN 7780
8165 GOTO 8190
8170 Y = INT (RND (1) * 300+1): IF
      Y > X THEN 6670
8175 GOTO 8190
8180 Y = INT (RND (1) * 400+1): IF
      Y > X THEN 9100
8190 PRINT "GOT 'EM!": LC(L1,L2) = -Q:
      EX(L1,L2) = -Q: IF Q = 3 THEN
      SV = SV+1
8192 IF Q = 4 THEN SV = SV + 3
8194 IF Q = 5 THEN SV = SV + 4
8196 IF Q = 6 THEN SV = SV + 2
8198 Q = -Q: GOTO 500
8199 REM * GUN *
8200 PRINT " ", "* BANG!": PRINT :DG =
      DG -2
8210 X = AX + (SX /2) + (PX / 5)
8220 IF Q = 3 THEN 8260
8230 IF Q = 4 THEN 8270
8240 IF Q = 5 THEN 8280
8250 Y = INT (RND (1) * 250 + 1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
8255 GOTO 8150
8260 Y = INT (RND (1) * 40 + 1): IF
      Y > X THEN 7780
8265 GOTO 8190
8270 Y = INT (RND (1) * 370 + 1): IF
      Y > X THEN 6670
8275 GOTO 8190
8280 Y = INT (RND (1) * 400 + 1): IF
      Y > X THEN 9100
```



```
8285 GOTO 8190
8299 REM * LASER *
8300 PRINT " ", "ZZZAP!!": PRINT :DG =
      DG - 1.5
8310 X = AX + (SX / 2) + (PX / 5)
8320 IF Q = 3 THEN 8360
8330 IF Q = 4 THEN 8370
8340 IF Q = 5 THEN 9100
8350 Y = INT (RND (1) * 300 + 1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
8355 GOTO 8150
8360 Y = INT (RND (1) * 230 + 1): IF
      Y > X THEN 7780
8365 GOTO 8190
8370 Y = INT (RND (1) * 300 + 1): IF
      Y > X THEN 6670
8375 GOTO 8190
8399 REM * ROPE *
8400 DG = DG - 15
8410 IF Q = 3 THEN 7780
8420 IF Q = 4 THEN 6670
8430 IF Q = 5 THEN 9100
8440 Y = INT (RND (1) * 250 + 1): IF
      Y < X THEN 8150
8450 PRINT " ", "GIBBLE!": GOTO 500
8499 REM * PIPE *
8500 DG = DG - 12: X = PX + SX + AX
8510 IF Q = 3 THEN 8560
8520 IF Q = 4 THEN 8570
8530 IF Q = 5 THEN 8570
8540 Y = INT (RND (1) * 400 + 1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
8550 GOTO 8150
8560 Y = INT (RND (1) * 350 + 1): IF
      Y > X THEN 7780
8565 GOTO 8190
8570 Y = INT (RND (1) * 500 + 1): IF
      Y > X THEN 6670
8575 GOTO 8190
```

```
8580 Y = INT (RND (1) * 700 + 1): IF
      Y > X THEN 9100
8585 GOTO 8190
8599 REM * ROCK *
8600 DG = DG - 12: X = PX + SX + (AX /
      3)
8610 IF Q = 3 THEN 8660
8620 IF Q = 4 THEN 6670
8630 IF Q = 5 THEN 8680
8640 PRINT " ", "GIBBLE!": GOTO 500
8660 Y = INT (RND (1) * 300 + 1): IF
      Y > X THEN 7780
8665 GOTO 8190
8680 Y = INT (RND (1) * 500 + 1): IF
      Y > X THEN 9100
8685 GOTO 8190
8699 REM * STICK *
8700 DG = DG - 15: X = SX + (AX / 2) +
      (PX / 2)
8710 IF Q = 3 THEN 8760
8720 IF (Q = 4) OR (Q = 5) THEN 8770
8730 Y = INT (RND (1) * 350 + 1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
8740 GOTO 8150
8760 Y = INT (RND (1) * 350 + 1): IF
      Y > X THEN 7780
8765 GOTO 8190
8770 PRINT "THE STICK SNAPS INTO
      PIECES.": DG = DG - 2: J(8) = 0
8780 IF Q = 4 THEN 6670
8785 GOTO 9100
8799 REM * GUM *
8800 DG = DG - 6: X = AX + (SX / 2) +
      (PX / 10)
8810 IF Q = 3 THEN 7780
8820 IF Q = 4 THEN 6670
8830 IF Q = 5 THEN 8860
8840 Y = INT (RND (1) * 260 + 1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
```

```

8850 GOTO 8150
8860 Y = INT (RND (1) * 200 + 1): IF
      Y > X THEN 9100
8865 GOTO 8190
8899 REM * SLIMY THING *
8900 IF Q = 3 THEN 8950
8910 IF Q = 4 THEN 6670
8920 IF Q = 5 THEN 8190
8930 PRINT " ", "GIBBLE!": GOTO 500
8950 PRINT "THE BRINCHLEY BEAST TURNS
      INTO A HUNGRY KUFU!"
8960 LC(L1,L2) = 4:EX(L1,L2) = 4:Q =
      4: GOTO 6670
8999 REM * SWORD *
9000 X = PX + (SX / 2) + (AX / 2):DG =
      DG - 5
9010 IF Q = 3 THEN 8190
9020 IF Q = 4 THEN 9050
9030 IF Q = 5 THEN 9070
9040 GOTO 8150
9050 Y = INT (RND (1) * 333 + 1): IF
      Y > X THEN 6670
9055 GOTO 8190
9070 Y = INT (RND (1) * 250 + 1): IF
      Y > X THEN 9100
9075 GOTO 8190

```

The Brinchley Beast attack routine appears at line 7780, the Kufu at 6670, and the Grimph at 9100. Purofolees do not attack. Programming for the various attack routines is listed in this chapter.

Assuming the requested weapon (in this case the knife) is available, the variable X is set to a value determined by the character's attributes—specifically, aim (AX), speed (SX) and power (PX):

```
8100 X = SX + PX + (AX/2): DG = DG-10
```

The exertion involved in using each weapon also results in a decrease of the current health rating (DG).

Each of the monsters will respond to each of the 10 weapons in different ways. To attack a Brinchley Beast with a knife, a random

number from 1 to 200 is selected (Y). If this number is greater than X, the Brinchley Beast will survive the attack and bite the explorer. If X happens to be greater than Y, the Brinchley Beast will be killed.

```
8110 IF Q = 3 THEN 8160
8160 Y = INT (RND (1)*200+1): IF Y>X
      THEN 7780
8165 GOTO 8190
8190 PRINT "GOT 'EM!": LC(L1,L2) = -Q:
      EX(L1,L2) = -Q: IF Q = 3 THEN
      SV = SV+1
8198 Q = -Q: GOTO 500
```

The Kufu is programmed in the same way, except Y is a random number from 1 to 300. This means there is a better chance of Y being greater than X. Therefore, a Kufu is harder to kill with a knife than a Brinchley Beast:

```
8120 IF Q = 4 THEN 8170
8170 Y = INT (RND (1) * 300+1): IF
      Y > X THEN 6670
8175 GOTO 8190
8192 IF Q = 4 THEN SV = SV + 3
```

The SV (monster killed score) value is increased by a value appropriate to the monster killed. Brinchley Beasts are worth 1, Purofolees 2, Kufus 3, and Grimphs 4. This is in line with how tough it is to kill each of these creatures.

The Grimph is the toughest to kill. For the knife, the Grimph's Y value may be as high as 400:

```
8130 IF Q = 5 THEN 8180
8180 Y = INT (RND (1) * 400+1): IF
      Y > X THEN 9100
8194 IF Q = 5 THEN SV = SV + 4
```

The Purofolee is programmed basically the same as the other monsters with one minor exception. Since the Purofolee does not attack, add a little personality by having it cry out. If it is killed it goes:

GIB-bleck...

If it survives your attack, it protests with an indignant:

GIBBLE!

For the knife, the Purofolee's maximum Y value is the same as the Brinchley Beast-200:

```

8140 Y = INT (RND (1) * 200+1): IF
      Y > X THEN PRINT " ", "GIBBLE!":
      GOTO 500
8150 PRINT " ", "GIB-BLECK...": GOTO
      8190
8196 IF Q = 6 THEN SV = SV + 2

```

The programming continues along the same basic lines for each of the nine other weapons with changes in the X and Y values. In some cases these values are not relevant. For instance, you'll never be able to kill a Brinchley Beast with a rope.

The effects of each weapon on each monster are summarized in Tables 4.2 through 4.5. The minimum value of X assumes the character values (PX, AX, and SX) were automatically generated. If manually entered values are used, it is possible to come up with lower values of X, making the monsters tougher to kill.

Table 4.2 Weapons Against Brinchley Beast.

Weapon	Y Max	X Min	Percent Success	X Max	Percent Success
1-knife	200	125	62.5	250	100
2-gun	90	85	94	170	100
3-laser	230	85	37	170	74
4-rope	-	-	0	-	0
5-pipe	350	150	43	300	86
6-rock	300	117	39	233	78
7-stick	350	100	28.5	200	57
8-gum	-	-	0	-	0
9-slimy thing	Brinchley Beast turns into a kufu				
10-sword	-	-	100	-	100

Table 4.3 Weapons Against Kufu.

Weapon	Y Max	X Min	Percent Success	X Max	Percent Success
1-knife	300	125	41.5	250	83
2-gun	370	85	23	170	46
3-laser	300	85	28	170	56.5
4-rope	—	—	0	—	0
5-pipe	500	150	30	300	60
6-rock	—	—	0	—	0
7-stick	—	—	0	—	0
8-gum	—	—	0	—	0
9-slimy thing	—	—	0	—	0
10-sword	333	100	30	200	60

Table 4.4 Weapons Against Grimph.

Weapon	Y Max	X Min	Percent Success	X Max	Percent Success
1-knife	400	125	31	250	62.5
2-gun	400	85	21	170	42.5
3-laser	—	—	0	—	0
4-rope	—	—	0	—	0
5-pipe	700	150	21	300	43
6-rock	500	117	23.5	233	46.5
7-stick	—	—	0	—	0
8-gum	200	80	40	160	80
9-slimy thing	—	—	100	—	100
10-sword	250	100	40	200	80

Table 4.5 Weapons Against Purofolee.

Weapon	Y Max	X Min	Percent Success	X Max	Percent Success
1-knife	200	125	62.5	250	100
2-gun	250	85	34	170	68
3-laser	300	85	28	170	56.5
4-rope	250	100	40	100	40
5-pipe	400	150	37.5	300	75
6-rock	—	—	0	—	0
7-stick	350	100	28.5	200	57
8-gum	260	80	31	160	61.5
9-slimy thing	—	—	0	—	0
10-sword	—	—	100	—	100

TOUCH COMMAND

The last command we will be adding to the MARS program is TOUCH. The programming for this command is in Listing 4.26.

Listing 4.26 Touch.

```
940 IF QX$ = "TOU" AND Q > 2 AND
    Q < 6 THEN 7700
7699 REM * TOUCH *
7700 IF Q = 3 THEN 7730
7710 PRINT "I WOULDN'T ADVISE THAT,";N$
7720 DG = DG - 5: GOTO 500
7730 IF TB = 1 THEN 7780
7740 PRINT "YOUR HAND FEELS RATHER
    STRANGE ..."
7750 FOR X = 1 TO 6: Y=INT (RND (1) *
    12 + 1): S(Y) = 1: A(Y) =0: B(Y) =
    0: NEXT
7760 DG = DG + (DG * .25): IF DG > DX
    THEN DG = DX
7770 TB = 1: GOTO 500
7780 X = INT (RND (1) * 7 + 1): PRINT
    "THE BRINCHLEY BEAST BITES YOUR ";
7790 IF X = 1 THEN PRINT "HAND!":DG =
    DG - 4
7800 IF X = 2 THEN PRINT "ARM!":DG =
    DG - 6
7810 IF X = 3 THEN PRINT "FOOT!":DG =
    DG - 5
7820 IF X = 4 THEN PRINT "TOE!":DG =
    DG - 2.5
7830 IF X = 5 THEN PRINT "ANKLE!":DG =
    DG - 5
7840 IF X = 6 THEN PRINT "POSTERIOR!":
    DG = DG - 8
7850 IF X = 7 THEN PRINT "KNEECAP!":DG
    = DG - 6
7860 GOTO 500
```

The TOUCH command is recognized only when a Brinchley Beast, Kufu, or Grimph is present:

```
940 IF QX$ = "TOU" AND Q > 2 AND  
    Q < 6 THEN 7700
```

but the command is only appropriate for a Brinchley Beast ($Q = 3$). If a player attempts to TOUCH a Kufu or Grimp, the computer will display a warning message, and 5 points will be subtracted from the current health rating (DG):

```
7700 IF Q = 3 THEN 7730  
7710 PRINT "I WOULDN'T ADVISE THAT,";N$  
7720 DG = DG - 5: GOTO 500
```

You should recall that when a Brinchley Beast is first encountered, the variable TB is set to 0. This is the Touch Beast counter. Touching a Brinchley Beast once is good, but touching it twice or more is not beneficial.

When the TOUCH command is entered when a Brinchley Beast is present, the value of TB is checked:

```
7730 IF TB = 1 THEN 7780
```

Assuming that TB is still set at 0, up to six supply items will be added to the explorer's inventory regardless of their previous location. (If the player already has the selected item, nothing will happen.)

This feature can come in handy for recovering lost items like inflatable raft (after it has been used), compass (after it has been broken by an angry Grimp), or food (after it has been eaten). Of course, the player is given no choice of which supplies the Brinchley Beast will endow on him:

```
7750 FOR X = 1 TO 6: Y=INT (RND (1) *  
    12 + 1): S(Y) = 1: A(Y) =0: B(Y) =  
    0: NEXT
```

The current health rating (DG) is also increased by up to 25 percent of its current value:

```
7760 DG = DG + (DG * .25): IF DG > DX  
    THEN DG = DX
```

Finally, TB is set to a value of 1:

7770 TB = 1: GOTO 500

On the second attempt to touch a Brinchley Beast, TB will equal 1, and instead of bestowing gifts, the creature will bite one of 7 randomly selected portions of the explorer's anatomy. This is done in lines 7780 through 7860. This is also the Brinchley Beast attack routine called from the KILL routine previously programmed. See also Tables 3.1 and 3.5.

SUMMARY

This is now a complete adventure game program. The new routines added in this chapter are summarized in Table 4.6. Table 4.7 lists the variables added in this chapter.

The things now missing from the MARS game are the instructions to be written in the next chapter.

**Table 4.6 Routines and Subroutines for the MARS Program
Presented in Chapter 4.**

Routines

500-620	determine obstacle present
1000-1070	move past monsters
5180-5290	CRY
5300-5680	EAT
5810-5890	DRINK at RIVER
5900-6090	FILL
6100-6190	INFLATE
6200-6380	CLIMB
6400-6530	OPEN BOX
6550-6660	PRAY
6670-6750	kufu attack
6760-6800	PRAY (continued)
7700-7770	TOUCH Brinchley Beast
7780-7860	Brinchley Beast attack
7900-9070	KILL
9100-9230	Grimph attack
9300-9330	Brinchley Beast attack
9350-9420	kufu present
9450-9460	Grimph present
9500-9570	purofolee present
9600-9680	River
9700-9810	move past River or Mountain

Subroutines

10850-10930	display dead monsters
10950-10960	Squeanly Serpent
11000-11120	Ghost
11900-12000	Ravine
12050-12160	Marsquake
12200-12270	storm
12300-12380	Funny colored sky
12400-12580	MAP
12600-12640	set up monster blocked moves

Table 4.7 Additional Variables for the MARS Program Added in Chapter 4.

MM	directional move allowed by monster
RV	river quality
TB	Touch Brinchley Beast counter
W	weapon used

See also Tables 3.1 and 3.5.

Chapter 5

Writing the Instructions

Once you've written an adventure game, add a set of instructions for the game so other people can play it. If it's any good at all, you'll want to share it sooner or later. Take pride in your creation.

You may put the game away for a while and play it again a few months later. You may well have forgotten the rules yourself by this time.

An instruction routine does not take up much memory space and is a worthwhile addition to any game program. Once in a while you may come up with a game so complex that it uses all of the memory by itself, so there is no room for instructions. This is the only circumstance when instructions should be left out. Even then, it is a good idea to write out the instructions and keep them with the tape or disk containing the program.

If it comes down to a choice between including instructions or adding special features like graphics or sound effects, opt for the instructions. Later, marketing and selling ideas for your game programs will be given. Commercial programs must *always* include clear instructions, since you cannot be there to tell the user all the things you forgot to include in the instructions.

PURPOSE

The first purpose of the instruction routine is to set up the game story. For the game of MARS we can start out by telling the player:

"You are on a historic mission to the planet Mars. A glorious civilization once thrived here. Your mission is to locate as many treasures as possible, return them to your space ship and blast off. But avoid excess junk weight. You will run into

many monsters and dangers while you explore the mysterious Martian landscape.”

Keep this introduction to the story simple and concise. Only tell the player what he needs to know. Consider the information contained in the introduction to MARS in Chapter 2. The player now knows the setting of the game (the planet Mars), the goal (recovery of ancient treasures), and the end of game condition (blasting off the space ship). The player is also told that there will be some junk among the treasures to avoid.

The last sentence about running into monsters and dangers isn't really necessary since they are an assumed and integral part of any adventure game. But the simple reminder in the introduction helps set up the atmosphere.

What you do not tell the player is as important as what you do tell him. Don't describe the monsters. And especially don't tell how to kill them! Let the player discover these things for himself. It's more fun.

However, you will need to tell the player what kind of responses will be expected of him. If the computer asks for a command, and the player has no idea of what to do, the game can fall flat.

MARS is written to accept one- or two-word commands (or one letter commands for directional moves). The instructions should make this information clear. It's a good idea to include a few examples.

We should also emphasize that only the first and last word are recognized in MARS. If the player enters GET GOLD AND CUP, the computer will treat the command as if it was just GET CUP. The gold will be ignored.

The full instruction routine for the MARS program is in Listing 5.1. Notice that it consists essentially of just PRINT commands. An instruction routine should not be at all complex.

In writing the instructions, carefully keep track of how much information is being displayed on the screen at any time. Obviously, no information should be allowed to scroll off the top of the screen before the player has a chance to read and digest it. Delay loops could be included after each full screen of information to allow time for reading. But different people read at different rates. The instructions might go by too fast for some players, and yet be deadly dull for others.

A better approach is to include a dummy INPUT statement (as in lines 10110, 10200, and 10330). This allows the player to determine

himself when the screen will be erased and new information displayed. Be sure to prompt the player to hit the 'RETURN' key. Don't assume that he will know what to do.

Once the program is finished, including the instructions, let several friends try the game. Tell them only what is included in the instructions displayed by the program. Anything they might have trouble with should be covered in the instructions. Rewrite the subroutine if necessary.

But try not to be *too* helpful in writing the instructions. If their trouble is "How the heck do I get rid of this damn Grimph?" that's just part of the game. Of course, if *nobody* can get past a Grimph, you may have made things too tough. Don't conclude that you just happen to have stupid friends and refuse to change anything. They probably represent other players well.

Listing 5.1 The Instructions Subroutine for the MARS Program.

```
10010 PRINT : PRINT "YOUR MISSION, ";N$; ",  
      IS TO EXPLORE THE"  
10020 PRINT "MYSTERIOUS PLANET MARS!":  
      PRINT  
10030 PRINT "AN ANCIENT CIVILIZATION  
      ONCE THRIVED"  
10035 PRINT "HERE. YOU MUST FIND  
      AS MANY OF THE"  
10040 PRINT "TREASURED RELICS  
      FROM THIS LONG-DEAD"  
10045 PRINT "CULTURE AS  
      YOU CAN, RETURN TO YOUR"  
10060 PRINT "ROCKET SHIP, AND BLAST  
      OFF FOR GOOD OLD EARTH.": PRINT  
10065 PRINT "BUT WATCH OUT -- SOME  
      ITEMS YOU WILL"  
10070 PRINT "FIND MAY BE  
      WORTHLESS JUNK THAT WILL"  
10075 PRINT "HAVE A NEGATIVE EFFECT  
      ON YOUR SCORE."  
10090 PRINT "SOME OF THE JUNK MIGHT,  
      HOWEVER, COME"  
10095 PRINT "IN HANDY DURING  
      YOUR EXPLORATION OF THE RED  
      PLANET.": PRINT
```

```
10110 INPUT "PLEASE PRESS 'RETURN' FOR  
      MORE      ";Q$  
10120 PRINTCHR$(147) : PRINT : PRINT  
      "YOU WILL ENCOUNTER A NUMBER OF  
      MARTIAN"  
10125 PRINT "BEASTS AND MONSTERS DURING YOUR  
      TRAVELS.";  
10130 PRINT "DIFFERENT TYPES OF  
      CREATURES MUST BE"  
10135 PRINT "HANDLED IN DIFFERENT WAYS. THE"  
10140 PRINT "SUCCESSFUL METHODS MAY NOT  
      ALWAYS BE OBVIOUS."  
10160 PRINT "THE BIZARRE NATURAL  
      FORCES OF MARS WILL"  
10165 PRINT "ALSO TEND TO  
      PRESENT OBSTACLES.": PRINT :  
      PRINT  
10170 PRINT "YOU MAY DEFINE THE  
      CHARACTERISTICS OF"  
10175 PRINT "YOUR EXPLORER BY SETTING A  
      VALUE FROM"  
10180 PRINT "1 TO 100%  
      FOR EACH QUALITY (SUCH AS"  
10185 PRINT "STRENGTH, SPEED, ETC.), OR YOU  
      CAN LET"  
10200 PRINT "THE COMPUTER AUTOMATICALLY  
      DEFINE YOUR CHARACTER FOR YOU."  
10205 PRINT : INPUT "PLEASE PRESS  
      'RETURN' FOR MORE      ";Q$  
10210 PRINTCHR$(147) : PRINT : PRINT  
      "DURING THE GAME, EACH COMMAND MAY BE"  
10215 PRINT "ONE OR TWO WORDS, SUCH AS  
      -- LOOK,"  
10216 PRINT "DROP ROCK, EAT FOOD, WAIT"  
10230 PRINT "OR A SINGLE LETTER  
      DIRECTIONAL MOVE"  
10235 PRINT "COMMAND:  
      (N=NORTH,S=SOUTH,E=EAST,"  
10240 PRINT "W=WEST).  COMMANDS  
      CONSISTING OF MORE THAN TWO WORDS,  
      SUCH AS:"
```

```
10260 PRINT "'PUT KUFU IN POCKET' ARE
      NOT VALID.": PRINT
10265 PRINT "IF YOU HAVE NO IDEA WHAT TO
      DO, YOU MAY"
10270 PRINT "USE THE SPECIAL COMMAND
      'HELP' FOR A"
10275 PRINT "LIST OF COMMANDS USED IN
      THE GAME."
10290 PRINT "NOT ALL COMMANDS WILL BE
      ACCEPTABLE"
10295 PRINT "UNDER ALL CIRCUMSTANCES!
      ALSO, WHAT"
10300 PRINT "MAY BE HELPFUL AT ONE
      TIME, MAY BE OF"
10310 PRINT "NO PARTICULAR EFFECT
      AT ANOTHER, AND"
10320 PRINT "ACTUALLY HARMFUL AT A THIRD
      TIME. FOR"
10325 PRINT "INSTANCE, EATING IN FRONT
      OF A GRIMPH IS NOT RECOMMENDED.":
      PRINT
10330 INPUT "PLEASE PRESS 'RETURN' TO
      DEFINE YOUR CHARACTER      ";Q$
10335 PRINTCHR$(147)
10340 RETURN
```

In conclusion, make the instructions clear enough so it is possible to win the game, but not so clear as to destroy the point of the game. If everybody wins, or if everybody loses, your program has problems.

Chapter 6

The Complete "MARS" Program

We have now completed the programming for the game of MARS. It is rather lengthy and will not fit into a computer with 16K of memory without a considerable amount of editing. However, a machine with 32K or 48K will have plenty of memory.

Listing 6.1 shows a printout for the complete MARS program.

In Chapters 8-10, three additional game programs will be presented and discussed. These programs will not be analyzed in as much depth as MARS was in the previous chapters. But they offer a good overview of the various techniques and methods of writing computerized adventure games.

Have fun!

Listing 6.1 Complete MARS Program.

```
1 REM * MARS * DELTON T. HORN * V1.0
5 DIMA(12),B(12),C(12),D(12),E(12),F(12)
10 DIMS(12),J(12),T(12),LC(10,10),EX(10,
10)
15 CL$=CHR$(147):REM SETUP CLEAR SCREEN
20 PRINTCL$," M A R S":PRINT:INPUT"YOUR
NAME";N$
30 PRINT:INPUT"WILL YOU NEED INSTRUCTION
S";Q$:Q$=LEFT$(Q$,1)
35 IFQ$="Y"THENGOSUB10010
36 PRINTCL$
40 FORX=1TO10:FORY=1TO10:EX(X,Y)=0
50 PRINT" * ";Z=INT(RND(1)*17+1):IFZ>1
2THENZ=0
```

```

60 LC(X,Y)=Z:NEXT:NEXT
70 PRINT"      PLEASE BE PATIENT, ";N$
80 R1=INT(RND(1)*10+1):R2=INT(RND(1)*10+
1):L1=R1:L2=R2
90 EX(R1,R2)=20:LC(R1,R2)=20:PRINT:PRINT
"I'M BUILDING AN ENTIRE PLANET HERE!"
100 PRINT:FORX=1TO12:A(X)=R1:B(X)=R2
110 Y=INT(RND(1)*10+1):Z=INT(RND(1)*10+1
):IFY=R1ANDZ=R2THEN110
120 C(X)=Y:D(X)=Z
130 Y=INT(RND(1)*10+1):Z=INT(RND(1)*10+1
):IFY=R1ANDZ=R2THEN130
140 E(X)=Y:F(X)=Z
150 S(X)=0:J(X)=0:T(X)=0
160 NEXT:GOSUB10000:INPUT"PLEASE PRESS '
RETURN' ";Q$
170 PRINT CL$
175 PRINT
180 PRINT"ENTER 1 FOR AUTOMATIC CHARACTE
R OR 2 TO CREATE YOUR OWN":INPUT X
190 IFX=1THEN210
200 IFX=2THEN230
205 GOTO180
210 AX=INT(RND(1)*50+1)+50:DX=INT(RND(1)
*100+1)+100
215 SX=INT(RND(1)*50+1)+50:PX=INT(RND(1)
*50+1)+50
220 GOTO300
230 INPUT"HEALTH";DX:IFDX<10ORDX>100THEN2
30
240 DX=DX*2:INPUT"SPEED";SX:IFSX<10ORSX>1
00THEN240
250 INPUT"POWER";PX:IFPX<10RPX>100THEN25
0
260 INPUT"AIM";AX:IFAX<10RAX>100THEN260
300 PRINTCHR$(147):PRINT:PRINT" ",N$:PRI
NT
310 PRINT"HEALTH",DX/2;"%"
320 PRINT"SPEED",SX;"%"
330 PRINT"POWER",PX;"%"
340 PRINT"AIM",AX;"%"

```

```
370 DG=DX
380 INPUT"PLEASE PRESS 'RETURN' TO PLAY
    ";Q$:PRINT CL$:PRINT:PRINT
399 REM * LOCATION DISPLAY *
400 Q=LC(L1,L2):PRINT:PRINT"    YOUR CURR
ENT COORDINATES ARE ";
405 GOSUB12600
410 IFS(11)=0THENPRINT"?:?:GOTO420
415 PRINTL1;" ";L2
420 IFL1=R1ANDL2=R2THEN430
425 IFS(12)=0THEN5000
430 IFL1=R1ANDL2=R2THENPRINT"YOU ARE SAF
ELY ABOARD YOUR ROCKET SHIP."
435 IFL1=R1ANDL2=R2THENDG=DG+3
440 Y=0:FORX=1TO12:IFA(X)=L1ANDB(X)=L2TH
ENGOSUB10450
450 IFC(X)=L1ANDD(X)=L2THENGOSUB10570
460 IFE(X)=L1ANDF(X)=L2THENGOSUB10700
470 IFY>8THENY=0:PRINT:INPUT"PLEASE PRES
S 'RETURN'    ";Q$:PRINT
480 NEXTX
499 REM * CHECK FOR MONSTERS *
500 EX(L1,L2)=Q:IF(Q<0)AND(Q>-7)THENGOSU
B10850
510 IFQ=1THENGOSUB10950
520 IFQ=2THENGOSUB11000
530 IFQ=3THEN9300
540 IFQ=4THEN9350
550 IFQ=5THEN9450
560 IFQ=6THEN9500
570 IFQ=7THEN9600
580 IFQ=8THENPRINT"YOU ARE AT THE FOOT O
F A TALL, CRAGGY MOUNTAIN"
590 IFQ=9THENGOSUB11900
600 IFQ=10THENGOSUB12050
610 IFQ=11THENGOSUB12200
620 IFQ=12THENGOSUB12300
649 REM * HEALTH CHECK *
650 IFDG<1THEN5070
660 IFDG<40THENPRINT"YOU'RE NOT LOOKING
TOO WELL, PAL."
```

```
699 REM * MAIN COMMAND *
700 Q$="":DG=DG-1:PRINT:PRINT"YOUR COMMA
ND, ";N$
710 INPUTQ$
715 IF (Q>2) AND (Q<7) THEN1000
717 IF (Q=7ORQ=8) AND (Q$="N"ORQ$="S"ORQ$="
E"ORQ$="W") THEN9700
720 IFQ$="N" THEN5100
730 IFQ$="S" THEN5120
740 IFQ$="E" THEN5140
750 IFQ$="W" THEN5160
760 QX$=LEFT$(Q$,3):QY$=RIGHT$(Q$,3)
770 IFQX$="SC0" THENGOSUB10400:PRINT"YOUR
SCORE SO FAR IS ";SC:GOTO700
780 IFQX$="DIA" THENGOSUB11150:GOTO700
790 IFQX$="CRY" THEN5180
800 IFQX$="EAT" THEN5300
810 IFQX$="DRI" THEN5700
820 IFQ=7ANDQX$="FIL" THEN5900
830 IFS(7)=1ANDQX$="INF" THEN6100
840 IFQX$="INV" THENGOSUB11300:GOTO700
850 IFQ=8ANDQX$="CLI" THEN6200
860 IFQX$="LOO" THEN400
870 IFT(9)=1ANDQ$="OPEN BOX" THEN6400
880 IFQX$="PRA" THEN6550
890 IFQX$="GET" THEN6810
900 IFQX$="DRO" THEN6920
910 IFQX$="KIL" THEN7900
920 IFQX$="HEL" THENGOSUB11800:GOTO700
930 IFQX$="WAI" THEN9820
940 IFQX$="TOU" ANDQ>2ANDQ<6 THEN7700
950 IFQX$="MAP" THENGOSUB12400:GOTO700
960 IFQX$="BLA" THEN9960
970 GOSUB11700
980 DG=DG-1:GOTO500
999 REM * MONSTER BLOCK MOVE *
1000 IFQ$="N"ORQ$="S"ORQ$="W"ORQ$="E" THE
N1010
1005 GOTO760
1010 IFQ$="N" ANDMM=1 THEN5100
1015 IFQ$="S" ANDMM=2 THEN5120
1020 IFQ$="E" ANDMM=3 THEN5140
```

```
1025 IFQ$="W"ANDMM=4THEN5160
1030 IFQ=5THEN9100
1040 IFQ=3THENPRINT"THE BRINCHLEY BEAST
BLOCKS YOUR PATH."
1050 IFQ=4THENPRINT"THE KUFU WILL NOT LE
T YOU GO THAT WAY.":DG=DG-2
1060 IFQ=6THENPRINT"THE DURN PUROFOLEE I
S IN YOUR WAY!"
1070 GOTO650
4999 STOP
5000 PRINT"YOU ARE OUTSIDE WITHOUT YOUR
SPACESUIT!":PRINT
5010 FORX=1TO4:Y=INT(RND(1)*5+1):Z=INT(R
ND(1)*75+1)+25:FORZZ=1TOZ:NEXT
5020 IFY=1THENPRINT"* GASP *",
5030 IFY=2THENPRINT"* CHOKE *",
5040 IFY=3THENPRINT"* PANT-PANT*",
5050 IFY=4THENPRINT"* WHEEZE *",
5060 NEXTX:PRINT
5070 GOSUB10000:PRINT:PRINT"YOU ARE DECE
ASED.":PRINT
5080 PRINT"YOUR FINAL SCORE WAS ":
5090 GOSUB10400:PRINTSC
5092 K$="LOF$,8"+CHR$(13)+"RU"+CHR$(13)
:F$="MENU":FORI=631TO640
5094 POKE I,ASC(MID$(K$,I-630,1)):NEXTI
:POKE 198,10:END
5099 REM * MOVES * NORTH *
5100 K=0:L3=L1:L4=L2:L1=L1-1:IFL1<1THENL
1=10
5110 GOTO400
5119 REM * SOUTH *
5120 K=0:L3=L1:L4=L2:L1=L1+1:IFL1>10THEN
L1=1
5130 GOTO400
5139 REM * EAST *
5140 K=0:L3=L1:L4=L2:L2=L2+1:IFL2>10THEN
L2=1
5150 GOTO400
5159 REM * WEST *
5160 K=0:L3=L1:L4=L2:L2=L2-1:IFL2<1THENL
2=10
```

```
5170 GOTO400
5179 REM * CRY *
5180 IFQ=4THEN5220
5190 IFQ=6THEN5240
5200 PRINT"WHY? ARE YOU UPSET FOR SOME
REASON?"
5210 GOTO500
5220 PRINT"KUFUS AREN'T NOTED FOR BEING
SOFT-HEARTED."
5230 GOTO500
5240 X=INT(RND(1)*10+1):IFX>5THEN5270
5260 PRINT"ANNOYED BY THE NOISE YOU'RE M
AKING, THE"
5265 PRINT"PUROFOLEE KNOCKS YOU DOWN AND
JUMPS ON YOUR STOMACH 17 TIMES!"
5266 DG = DG-17:GOTO 500
5270 PRINT"FRIGHTENED BY THE NOISE YOU'R
E MAKING, THE PUROFOLEE HOPS OFF."
5280 X=L1+INT(RND(1)*5+1)-3:Y=L2+INT(RND
(1)*5+1)-3:IFX=L1ANDY=L2THEN5280
5285 IFX=R1ANDY=R2THEN5280
5290 LC(L1,L2)=0:EX(L1,L2)=0:Q=0:LC(X,Y)
=6:GOTO500
5299 REM * EAT *
5300 IFQ>3ANDQ<7THEN5600
5310 IFQY$="OOD"THEN5420
5320 IFQY$="UFU"THEN5480
5330 IFQY$="LEE"THEN5540
5340 X=LEN(Q$):IFX<5THEN5590
5350 X=INT(RND(1)*6+1):DG=DG-.5:IFX=1THE
NPRINT"EAT WHAT???"
5360 IFX=2THENPRINT"-ER- NO, THANK YOU..
."
5370 IFX=3THENPRINT"ARE YOU NUTS?"
5380 IFX=4THENPRINT"THAT HUNGRY, I'M NOT
!"
5390 IFX=5THENPRINT"YUK!"
5400 IFX=6THENPRINT"HAS ANYONE EVER TOLD
YOU THAT YOU HAVE WEIRD TASTES?"
5410 GOTO500
5420 IFS(1)=1THEN5450
```

```
5430 PRINT"YOU DON'T HAVE ANY!"
5440 DG=DG-1:GOTO500
5450 PRINT" ", "BURP":PRINT
5460 DG=DG+(DX*.25):IFDG>DXTHENDG=DX
5470 S(1)=0:GOTO500
5480 IFQ=-4THEN5500
5490 PRINT"FIRST YOU HAVE TO GO OUT AND
KILL ONE, ";N$:DG=DG-1:GOTO500
5500 PRINT"YOU HAVE TO HOLD YOUR NOSE TO
GET CLOSE ENOUGH,"
5510 DG=DG+INT(RND(1)*15+1)-10:IFDG>DXTH
ENDG=DX
5520 PRINT"BUT YOU SOMEHOW MANAGE TO GET
IT DOWN.":LC(L1,L2)=0
5530 EX(L1,L2)=0:Q=0:GOTO500
5540 IFQ=-6THEN5560
5550 GOTO5490
5560 PRINT" ", "YUMMY!":PRINT:DG=DG+50
5570 IFDG>DXTHENDG=DX
5580 LC(L1,L2)=0:EX(L1,L2)=0:Q=0:GOTO500
5590 PRINT"EAT WHAT?":DG=DG-.25:GOTO500
5600 IFQ=4THEN5640
5610 IFQ=5THEN5670
5620 IFS(1)=0THENPRINT"THERE IS NOTHING
HERE TO EAT.":DG=DG-1:GOTO500
5630 PRINT"WHEN YOU TAKE YOUR FOOD OUT,
THE DURN "
5635 PRINT"PUROFOLEE SNATCHES IT BEFORE
YOU CAN TAKE A SINGLE BITE!"
5636 PRINT:S(1)=0:DG=DG-1:GOTO500
5640 PRINT"THIS IS NO TIME TO THINK OF Y
OUR STOMACH, ";N$;" !"
5650 GOSUB10000:PRINT"UNLESS YOU WANT TO
SEE THE INSIDE OF A KUFU'S STOMACH!"
5660 DG=DG-2:GOTO500
5670 PRINT"THE GRIMPH THINKS THAT'S A JI
M DANDY IDEA!":PRINT:GOSUB10000
5680 PRINT"IT EATS YOU!":PRINT:GOTO5070
5699 REM * DRINK *
5700 IFQ=7THEN5810
5710 IFS(2)=1THEN5740
```



```
5720 IFS(2)=2THEN5800
5730 PRINT"THERE IS NOTHING HERE TO DRIN
K, ";N$:DG=DG-.5:GOTO500
5740 PRINT" ",:FORX=1TO3:FORY=1TO85:NEXT
Y
5750 PRINT"* GLUG * ",:NEXTX:PRINT
5760 GOSUB10000
5770 PRINT" ", "AHHHH!":DG=DG+(DX*.15)
5780 IFDG>DXTHENDG=DX
5790 S(2)=2:GOTO500
5800 PRINT"YOUR WATER BOTTLE IS EMPTY.":
DG=DG-.4:GOTO500
5810 PRINT" ", "SLURP!!!":PRINT:GOSUB1000
0
5820 X=INT(RND(1)*10+1):IFX>7THEN5860
5830 IFX<4THEN5870
5840 DG=DG+DX*.15:IFDG>DXTHENDG=DX
5850 GOTO500
5860 PRINT"IT SURE TASTES GOOD!":DG=DG+D
X*.1:GOTO5840
5870 PRINT"THIS WATER MAKES YOU QUITE IL
L.":DG=DG*.75
5880 PRINT"BUT YOU DON'T DIE ";:GOSUB100
00
5890 PRINT"--- AT LEAST, NOT JUST YET.":
GOTO500
5899 REM * FILL *
5900 X=LEN(Q$):IFX>6THEN5920
5910 PRINT"FILL WHAT?":DG=DG-.5:GOTO500
5920 IFQY$="TLE"THEN5990
5930 IFQY$="URN"THEN6040
5940 IFQY$="BOX"THEN6060
5950 IFQY$="UIT"THENPRINT"YOU DROWN!":GO
TO5070
5960 IFQY$="OWL"THEN6080
5970 IFQY$="CUP"THEN6090
5980 PRINT"THAT WON'T HOLD WATER, ";N$:D
G=DG-1:GOTO500
5990 IFS(2)=2THEN6030
6000 IFS(2)=1THEN6020
6010 PRINT"YOU DON'T HAVE IT, ";N$;"!":D
G=DG-1.25:GOTO500
```

```
6020 PRINT"IT'S ALREADY FULL":DG=DG-1:GO
    TO500
6030 PRINT"OK.  THE BOTTLE IS NOW FULL."
    :S(2)=1:GOTO500
6040 IFJ(8)=0THEN6010
6045 PRINT"THE URN";
6050 PRINT" LEAKS. ":DG=DG-.75:GOTO500
6060 IFT(9)=0THEN6010
6070 PRINT"THE BOX DISSOLVES.":T(9)=0:GO
    TO500
6080 IFT(1)=0THEN6010
6085 PRINT"THE COPPER BOWL":GOTO6050
6090 IFT(5)=0THEN6010
6095 PRINT"THE SILVER CUP":GOTO6050
6099 REM * INFLATE RAFT *
6100 PRINT" ", "PHEFFTT!":PRINT
6110 GOSUB10000:IFQ=7THEN6130
6120 PRINT"THE RAFT SPRINGS A LEAK!":PRI
    NT:S(7)=0:GOTO500
6130 DG=DG-(INT(RND(1)*50+1)/10)
6140 PRINT"YOU CAN NOT CONTROL THE RAFT
    ON THESE RAGING CURRENTS"
6150 PRINT:GOSUB10000
6160 PRINT"YOU MAKE IT TO A SHORE, BUT Y
    OU LOSE YOUR RAFT!"
6170 X=L1-2+INT(RND(1)*3+1):Y=L2-2+INT(R
    ND(1)*3+1):IFX=L1ANDY=L2THEN6170
6172 IF X<1 THEN X=10
6174 IF X>10 THEN X=1
6176 IF Y<1 THEN Y=10
6178 IF Y>10 THEN Y=1
6180 S(7)=0:L3=L1:L4=L2:L1=X:L2=Y:INPUT"
    PLEASE PRESS 'RETURN' ";Q$
6190 GOTO400
6199 REM * CLIMB MOUNTAIN *
6200 INPUT"DIRECTION":Q$:Q$=LEFT$(Q$,1)
6210 IF(Q$="N")OR(Q$="S")OR(Q$="E")OR(Q$
    ="W")THEN6230
6220 GOTO6200
6230 IFS(6)=1THEN6270
6240 X=INT(RND(1)*10+1)
6250 IFJ(12)=1THEN6320
```

```
6260 IFX>7THEN720
6265 GOTO6360
6270 INPUT"DO YOU USE YOUR ROPE";X$:X$=L
EFT$(X$,1)
6280 IFX$="Y"THEN6300
6290 GOTO6240
6300 X=INT(RND(1)*20+1):IFJ(12)=1THEN632
0
6310 GOTO6260
6320 Y=INT(RND(1)*12+1):IFY<5THEN6260
6330 IFS(12)=0THEN6360
6335 PRINT"HALF WAY UP THE MOUNTAIN, YOU
DROP THE SLIMY THING!"
6340 J(12)=0:C(12)=L1:D(12)=L2:IFY>8THEN
6260
6350 PRINT"YOU TRIP ON THE SLIMY THING A
ND FALL!":GOTO6370
6360 PRINT"YOU FALL!"
6370 DG=DG-INT(RND(1)*DG+1)+1
6380 GOTO500
6399 REM * OPEN BOX *
6400 PRINT:PRINT" ","HMMM....":PRINT:GO
SUB10000
6410 IFQ=6THEN6470
6420 IFQ=4THEN6500
6430 FORX=1TO3:Y=INT(RND(1)*12+1):IFJ(Y)
=1THENJ(Y)=0
6440 NEXT:IFQX$="PRA"THEN500
6450 PRINT"THE BOX SNAPS SHUT!":PRINT" "
,"CLICK":PRINT
6460 GOTO500
6470 PRINT:PRINT" ","*** POOF ***":PRINT
6480 GOSUB10000:Q=0:LC(L1,L2)=0:EX(L1,L2
)=0
6490 PRINT"THE PUROFOLEE VANISHES.":PRIN
T:GOSUB10000:GOTO6450
6500 PRINT" ","*** POOF ***":PRINT:GOSUB
10000
6510 LC(L1,L2)=5:EX(L1,L2)=5:Q=5
6520 PRINT"THE KUFU JUST TURNED INTO A G
RIMPH!"
6530 GOTO500
```

```
6550 IFQ=3THEN6600
6560 IFQ=4THEN6650
6570 IFQ=5THEN6760
6580 IFQ=6THEN6800
6590 DG=DG+1:PRINT:PRINT"GIMME THAT OLD
TIME RELIGION!":PRINT:GOTO500
6600 IFT(4)=1THEN6620
6610 PRINT"NOTHING MUCH SEEMS TO HAPPEN.
":DG=DG+1:GOTO500
6620 PRINT:PRINT" ","*** POOF ***":PRINT
:GOSUB10000
6630 PRINT"THE BRINCHLEY BEAST TURNS INT
O A KUFU!"
6640 LC(L1,L2)=4:EX(L1,L2)=4:Q=4:GOTO500
6650 PRINT"KUFUS AREN'T KNOWN FOR BEING
PIOUS":PRINT:GOSUB10000
6660 PRINT"IT TAKES ADVANTAGE OF YOUR IN
ATTENTION TO ATTACK!":PRINT
6670 PRINT"IT BITES YOUR ";X=INT(RND(1)
*7+1)
6680 IFX=1THENPRINT"ARM!":DG=DG-5
6690 IFX=2THENPRINT"LEG!":DG=DG-7
6700 IFX=3THENPRINT"STOMACH!":DG=DG-50
6710 IFX=4THENPRINT"NECK!":DG=DG-75
6720 IFX=5THENPRINT"AIR HOSE!":GOTO5070
6730 IFX=6THENPRINT"NUSE!":DG=DG-20
6740 IFX=7THENPRINT"POSTERIOR!":DG=DG-30
6750 GOTO650
6760 PRINT:PRINT"SORRY --- ":PRINT:GOSUB
10000
6770 PRINT"ALL MARTIAN DEITIES ARE BUSY
AT THE      MOMENT.":GOSUB10000
6780 PRINT"PLEASE CALL AGAIN.":PRINT:GOS
UB10000
6790 PRINT" ","HAVE A NICE DAY!!":PRINT:
PRINT:GOTO500
6800 IFS(5)=1THEN6760
6805 GOTO6470
6809 REM * GET/DROP *
6810 IFQ$="GET"THENPRINT"GET WHAT?":GOTO
500
6815 IFQ$="GET ALL"THEN9900
```

```
6820 IFQY$="OST"THEN6870
6830 IFQ$="GET SICK"THEN6880
6840 IFQ$="ENT"ORQ$="KED"THEN6890
6850 IFQY$="OWN"THEN6900
6860 G=1:Y=0:FORX=1TO12:Y=Y+S(X)+J(X)+I(X):NEXT:IFY>17THEN9850
6865 GOT06930
6870 PRINT"I THINK YOU ALREADY ARE...":G
    OT0500
6880 PRINT"THAT'S DISGUSTING!":GOT0500
6890 PRINT"SAME TO YOU, ";N$; "!":GOT0500
6900 PRINT"THIS IS NO TIME TO BOOGIE!":G
    OT0500
6910 IFQY$="EAD"THEN6890
6920 G=2
6930 IFQY$="OOD"THENU=1:GOT07500
6940 IFQY$="TLE"ORQY$="TER"THENU=2:GOT07
    500
6950 IFQY$="IFE"THENU=3:GOT07500
6960 IFQY$="GUN"THENU=4:GOT07500
6970 IFQY$="SER"THENU=5:GOT07500
6980 IFQY$="OPE"ORQY$="OIL"THENU=6:GOT07
    500
6990 IFQY$="AFT"THENU=7:GOT07500
7000 IFQY$="GHT"THENU=8:GOT07500
7010 IFQY$="IFE"THENU=9:GOT07500
7020 IFRIGHT$(Q$,4)="INES"THENU=10:GOT07
    500
7030 IFQY$="ASS"THENU=11:GOT07500
7040 IFQY$="UIT"ANDG=2THEN5070
7050 IFQY$="UIT"THENU=12:GOT07500
7060 IFQY$="HDE"THENU=1:GOT07600
7070 IFRIGHT$(Q$,4)="RING"THENU=2:GOT076
    00
7080 IFQY$="OCK"THENU=3:GOT07600
7090 IFQY$="RTS"THENU=4:GOT07600
7100 IFQY$="IRT"ORQY$="LOT"THENU=5:GOT07
    600
7110 IFQY$="ONE"THENU=6:GOT07600
7120 IFQY$="ICK"THENU=7:GOT07600
```

```
7130 IFQY$="URN"THENU=8:GOTO7600
7140 IFQY$="WAD"ORQY$="GUM"THENU=9:GOTO7
600
7150 IFQY$="WER"THENU=10:GOTO7600
7160 IFQY$="FLY"THENU=11:GOTO7600
7170 IFRIGHT$(Q$,4)="HING"THENU=12:GOTO7
600
7180 IFQY$="PER"ORQY$="OWL"THENU=1:GOTO7
650
7190 IFQY$="INS"THENU=2:GOTO7650
7200 IFQY$="ULE"THENU=3:GOTO7650
7210 IFQY$="TUE"ORQY$="GOD"THENU=4:GOTO7
650
7220 IFQY$="VER"ORQY$="CUP"THENU=5:GOTO7
650
7230 IFQY$="ORB"THENU=6:GOTO7650
7240 IFQY$="OLL"THENU=7:GOTO7650
7250 IFRIGHT$(Q$,4)="ONES"THENU=8:GOTO76
50
7260 IFQY$="BOX"THENU=9:GOTO7650
7270 IFQY$="ORD"THENU=10:GOTO7650
7280 IFQY$="ULL"THENU=11:GOTO7650
7290 IFQY$="NTS"THENU=12:GOTO7650
7300 X=LEN(Q$):IFX<6THEN7350
7310 IFG=1THENX=X-4:GOTO7320
7315 X=X-5
7320 QY$=RIGHT$(Q$,X)
7330 PRINT"SORRY, ";N$;", BUT I DON'T SE
E ANY":PRINTQY$;" HERE."
7350 PRINT"PLEASE TRY TO KEEP YOUR COMMA
NDS          RATIONAL IN THE FUTURE."
7360 PRINT:DG=DG-3:GOTO500
7500 IFG=2THEN7550
7510 IFS(U)>0THEN7540
7515 IFA(U)=L1ANDB(U)=L2THEN7520
7516 GOTO7590
7520 S(U)=1:A(U)=0:B(U)=0:PRINT"OK":GOTO
500
7540 PRINT"YOU ALREADY HAVE IT!":DG=DG-1
:GOTO500
```

```
7550 IFS(U)<1THEN7580
7560 S(U)=0:A(U)=L1:B(U)=L2:PRINT"OK":GO
T0500
7580 PRINT"YOU DON'T HAVE IT!":DG=DG-1:G
OT0500
7590 PRINT"IT'S NOT HERE, ";N$:DG=DG-1:G
OT0500
7600 IFG=2THEN7630
7610 IFJ(U)>0THEN7540
7615 IFC(U)=L1ANDD(U)=L2THEN7620
7617 GOT07590
7620 J(U)=1:C(U)=0:D(U)=0:PRINT"OK":GOTO
500
7630 IFJ(U)<1THEN7580
7640 J(U)=0:C(U)=L1:D(U)=L2:PRINT"OK":GO
T0500
7650 IFG=2THEN7680
7660 IFT(U)>0THEN7540
7665 IFE(U)=L1ANDF(U)=L2THEN7670
7667 GOT07590
7670 T(U)=1:E(U)=0:F(U)=0:PRINT"OK":GOTO
500
7680 IFT(U)<1THEN7580
7690 T(U)=0:E(U)=L1:F(U)=L2:PRINT"OK":GO
T0500
7699 REM * TOUCH *
7700 IFQ=3THEN7730
7710 PRINT"I WOULDN'T ADVISE THAT, ";N$
7720 DG=DG-5:GOT0500
7730 IFTB=1THEN7780
7740 PRINT"YOUR HAND FEELS RATHER STRANG
E ... "
7750 FORX=1T06:Y=INT(RND(1)*12+1):S(Y)=1
:A(Y)=0:B(Y)=0:NEXT
7760 DG=DG+(DG*.25):IFDG>DXTHENDG=DX
7770 TB=1:GOT0500
7780 X=INT(RND(1)*7+1):PRINT"THE BRINCHL
EY BEAST BITES YOUR ";
7790 IFX=1THENPRINT"HAND!":DG=DG-4
7800 IFX=2THENPRINT"ARM!":DG=DG-6
7810 IFX=3THENPRINT"FOOT!":DG=DG-5
7820 IFX=4THENPRINT"TOE!":DG=DG-2.5
```

```
7830 IFX=5THENPRINT"ANKLE!":DG=DG-5
7840 IFX=6THENPRINT"POSTERIOR!":DG=DG-8
7850 IFX=7THENPRINT"KNEECAP!":DG=DG-6
7860 GOTO500
7899 REM * KILL *
7900 IF (Q>2)AND(Q<7)THEN7930
7910 PRINT"MY, BUT WE'RE IN A HOSTILE MO
OD TODAY!"
7920 DG=DG-2:GOTO500
7930 INPUT"YOUR CHOICE OF WEAPON":Q$
7940 IFQ$="KNIFE"THENW=1:GOTO8000
7945 IFQ$="GUN"THENW=2:GOTO8000
7950 IFQ$="LASER"THENW=3:GOTO8000
7955 QY$=RIGHT$(Q$,4):IFQY$="ROPE"ORQY$=
"COIL"THENW=4:GOTO8000
7960 IFQY$="PIPE"THENW=5:GOTO8000
7965 IFQY$="ROCK"THENW=6:GOTO8000
7970 IFQY$="TICK"THENW=7:GOTO8000
7975 IFQY$=" WAD"ORQY$=" GUM"THENW=8:GOT
O8000
7980 IFQY$="HING"THENW=9:GOTO8000
7985 IFQY$="WORD"THENW=10:GOTO8000
7990 PRINT"THAT DOESN'T SOUND LIKE A VER
Y GOOD WEAPON TO ME, ":N$
7995 GOTO500
8000 IFW=1ANDS(3)=1THEN8100
8005 IFW=2ANDS(4)=1THEN8200
8010 IFW=3ANDS(5)=1THEN8300
8015 IFW=4ANDS(6)=1THEN8400
8020 IFW=5ANDS(9)=1THEN8500
8025 IFW=6ANDJ(3)=1THEN8600
8030 IFW=7ANDJ(7)=1THEN8700
8035 IFW=8ANDJ(9)=1THEN8800
8040 IFW=9ANDJ(12)=1THEN8900
8045 IFW=10ANDT(10)=1THEN9000
8050 PRINT"YOU DON'T HAVE IT!!":PRINT
8060 IFQ=3THEN7780
8070 IFQ=4THENPRINT"THE KUFU IS PLEASED
BY YOUR ERROR!":GOTO6670
8080 IFQ=5THEN9100
8090 DG=DG-3:GOTO500
8099 REM * KNIFE *
```



```
8100 X= SX+PX+(AX/2):DG=DG-10
8110 IFQ=3THEN8160
8120 IFQ=4THEN8170
8130 IFQ=5THEN8180
8140 Y=INT(RND(1)*200+1):IFY>XTHENPRINT"
      ", "GIBBLE!":GOTO500
8150 PRINT" ", "GIB-BLECK...":GOTO8190
8160 Y=INT(RND(1)*200+1):IFY>XTHEN7780
8165 GOTO8190
8170 Y=INT(RND(1)*300+1):IFY>XTHEN6670
8175 GOTO8190
8180 Y=INT(RND(1)*400+1):IFY>XTHEN9100
8190 PRINT"GOT 'EM!":LC(L1,L2)=-Q:EX(L1,
L2)=-Q:IFQ=3THENSV=SV+1
8192 IFQ=4THENSV=SV+3
8194 IFQ=5THENSV=SV+4
8196 IFQ=6THENSV=SV+2
8198 Q=-Q:GOTO500
8199 REM * GUN *
8200 PRINT" ", "* BANG!":PRINT:DG=DG-2
8210 X=AX+(SX/2)+(PX/5)
8220 IFQ=3THEN8260
8230 IFQ=4THEN8270
8240 IFQ=5THEN8280
8250 Y=INT(RND(1)*250+1):IFY>XTHENPRINT"
      ", "GIBBLE!":GOTO500
8255 GOTO8150
8260 Y=INT(RND(1)*40+1):IFY>XTHEN7780
8265 GOTO8190
8270 Y=INT(RND(1)*370+1):IFY>XTHEN6670
8275 GOTO8190
8280 Y=INT(RND(1)*400+1):IFY>XTHEN9100
8285 GOTO8190
8299 REM * LASER *
8300 PRINT" ", "ZZZAP!!":PRINT:DG=DG-1.5
8310 X=AX+(SX/2)+(PX/5)
8320 IFQ=3THEN8360
8330 IFQ=4THEN8370
8340 IFQ=5THEN9100
8350 Y=INT(RND(1)*300+1):IFY>XTHENPRINT"
      ", "GIBBLE!":GOTO500
8355 GOTO8150
```

```
8360 Y=INT(RND(1)*230+1):IFY>XTHEN7780
8365 GOTO8190
8370 Y=INT(RND(1)*300+1):IFY>XTHEN6670
8375 GOTO8190
8399 REM * ROPE *
8400 DG=DG-15
8410 IFQ=3THEN7780
8420 IFQ=4THEN6670
8430 IFQ=5THEN9100
8440 Y=INT(RND(1)*250+1):IFY<XTHEN8150
8450 PRINT" ", "GIBBLE!":GOTO500
8499 REM * PIPE *
8500 DG=DG-12: X=PX+SX+AX
8510 IFQ=3THEN8560
8520 IFQ=4THEN8570
8530 IFQ=5THEN8570
8540 Y=INT(RND(1)*400+1):IFY>XTHENPRINT"
      ", "GIBBLE!":GOTO500
8550 GOTO8150
8560 Y=INT(RND(1)*350+1):IFY>XTHEN7780
8565 GOTO8190
8570 Y=INT(RND(1)*500+1):IFY>XTHEN6670
8575 GOTO8190
8580 Y=INT(RND(1)*700+1):IFY>XTHEN9100
8585 GOTO8190
8599 REM * ROCK *
8600 DG=DG-12: X=PX+SX+(AX/3)
8610 IFQ=3THEN8660
8620 IFQ=4THEN6670
8630 IFQ=5THEN8680
8640 PRINT" ", "GIBBLE!":GOTO500
8660 Y=INT(RND(1)*300+1):IFY>XTHEN7780
8665 GOTO8190
8680 Y=INT(RND(1)*500+1):IFY>XTHEN9100
8685 GOTO8190
8699 REM * STICK *
8700 DG=DG-15: X=SX+(AX/2)+(PX/2)
8710 IFQ=3THEN8760
8720 IF(Q=4)OR(Q=5)THEN8770
8730 Y=INT(RND(1)*350+1):IFY>XTHENPRINT"
      ", "GIBBLE!":GOTO500
8740 GOTO8150
```

```
8760 Y=INT(RND(1)*350+1):IFY>XTHEN7780
8765 GOTO8190
8770 PRINT"THE STICK SNAPS INTO PIECES."
      :DG=DG-2:J(8)=0
8780 IFQ=4THEN6670
8785 GOTO9100
8799 REM * GUM *
8800 DG=DG-6:X=AX+(SX/2)+(PX/10)
8810 IFQ=3THEN7780
8820 IFQ=4THEN6670
8830 IFQ=5THEN8860
8840 Y=INT(RND(1)*260+1):IFY>XTHENPRINT"
      ", "GIBBLE!":GOTO500
8850 GOTO8150
8860 Y=INT(RND(1)*200+1):IFY>XTHEN9100
8865 GOTO8190
8899 REM * SLIMY THING *
8900 IFQ=3THEN8950
8910 IFQ=4THEN6670
8920 IFQ=5THEN8190
8930 PRINT" ", "GIBBLE!":GOTO500
8950 PRINT"THE BRINCHLEY BEAST TURNS INT
O A HUNGRY KUFU!"
8960 LC(L1,L2)=4:EX(L1,L2)=4:Q=4:GOTO667
O
8999 REM * SWORD *
9000 X=PX+(SX/2)+(AX/2):DG=DG-5
9010 IFQ=3THEN8190
9020 IFQ=4THEN9050
9030 IFQ=5THEN9070
9040 GOTO8150
9050 Y=INT(RND(1)*333+1):IFY>XTHEN6670
9055 GOTO8190
9070 Y=INT(RND(1)*250+1):IFY>XTHEN9100
9075 GOTO8190
9100 PRINT"NONE TOO PLEASED WITH YOUR AT
TITUDE, ";
9110 PRINT"THE GRIMPH BREAKS YOUR ";
9120 X=INT(RND(1)*10+1):IFX=1ANDS(11)=0T
HEN9120
9130 IFX=1THENPRINT"COMPASS!":S(11)=0
9140 IFX=2THENPRINT"ARM!":DG=DG-15
```

```
9150 IFX=3THENPRINT"LEG!":DG=DG-20
9160 IFX=4THENPRINT"NECK!":DG=DG-70
9170 IFX=5THENPRINT"THUMBNAIL!":DG=DG-2
9180 IFX=6THENPRINT"NOSE!":DG=DG-10
9190 IFX=7THENPRINT"BIG TOE!":DG=DG-3
9200 IFX=8THENPRINT"BACK!":DG=DG-65
9210 IFX=9THENPRINT"FINGER!":DG=DG-3
9220 IFX=10THENPRINT"SKULL!":DG=DG-75
9230 GOTO650
9300 IFK=1THEN9330
9310 K=1:TB=0:PRINT" ", "BRINCH-":GOSUB1
0000
9320 PRINT"LEY!!!":PRINT:GOSUB10000
9330 PRINT"A BRINCHLEY BEAST IS HERE.":G
OTO650
9350 IFK=2THEN9400
9360 PRINT"A HUNGRY KUFU BLOCKS YOUR PAT
H!":GOSUB10000
9370 PRINT:PRINT"IT SALIVATES IN FOUL AN
TICIPATION OF THE MEAL TO COME!":PRINT
9380 K=2:GOTO9410
9400 PRINT"A KUFU IS HERE."
9410 X=INT(RND(1)*10+1):IFX>6THEN6670
9420 GOTO650
9450 PRINT"A GRIMPH IS HERE.":X=INT(RND(
1)*10+1):IFX>3THEN9110
9460 GOTO650
9500 IFK=3THEN9550
9510 GOSUB10000:FORX=1TO3:Z=INT(RND(1)*2
5+1)+1:FORY=1TOZ
9520 PRINT" ";:NEXTY:PRINT"GIBBLE! ";:N
EXTX
9530 PRINT:PRINT:PRINT"A WILD PUROFOLEE
HOPS INTO VIEW!":K=3
9540 GOTO650
9550 PRINT"A PUROFOLEE IS HERE.":PRINT
9560 PRINT"PUROFOLEE: GIBBLE?":PRINT
9570 GOTO650
9600 IFK=4THEN9650
9610 RV=INT(RND(1)*5+1):PRINT"YOU COME T
O A RIVER BANK"
9620 PRINT:K=4:GOTO9660
```

```
9650 PRINT"YOU ARE AT THE BANK OF A RIVE
R.":PRINT
9660 IFRV=1THENPRINT"IT IS QUITE PLEASAN
T HERE."
9670 IFRV=3THENPRINT"THE SMELL OF ANCIEN
T SEWAGE IS"
9675 IFRV=3THENPRINT"UNPLEASANT, BUT BEA
RABLE."
9680 DG=DG+.25:GOTO650
9700 X=L1:Y=L2:IFQ$="N"THENX=L1-1
9710 IFX<1THENX=10
9720 IFQ$="S"THENX=L1+1
9730 IFX>10THENX=1
9740 IFQ$="E"THENY=L2+1
9750 IFY>10THENY=1
9760 IFQ$="W"THENY=L2-1
9770 IFY<1THENY=10
9780 IFX=L3ANDY=L4THEN9810
9790 IFQ=7THENPRINT"THE RIVER ";:GOTO980
0
9795 PRINT"THE MOUNTAIN ";
9800 PRINT"IS IN YOUR WAY.":DG=DG-3:GOTO
650
9810 L3=L1:L4=L2:L1=X:L2=Y:K=0:GOTO400
9819 REM * WAIT *
9820 PRINT" ", "(PAUSE)":PRINT:GOSUB10000
9830 DG=DG+10:IFDG>DXTHENDG=DX
9840 GOTO500
9850 PRINT"YOUR ARMS ARE FULL. YOU CAN'
T CARRY"
9855 PRINT"ANYTHING MORE UNLESS YOU DROP
SOMETHING."
9860 DG=DG-1:GOTO500
9899 REM * GET ALL *
9900 IFL1=R1ANDL2=R2THEN9920
9910 GOTO970
9920 FORX=1TO12:IFA(X)=L1ANDB(X)=L2THEN9
950
9930 NEXT:PRINT"SUPPLIES GATHERED.":GOTO
500
9950 A(X)=0:B(X)=0:S(X)=1:GOTO9930
9960 IFL1=R1ANDL2=R2THEN9980
```

```
9970 PRINT"YOU DON'T HAVE JET PROPULSION
    ENGINES IN"
9975 PRINT"YOUR SHOES!":DG=DG-2:GOTO500
9980 FORX=1TO100:PRINT"  *  ";:FORY=1TO5
5:NEXT:NEXT
9985 PRINT:PRINT:GOSUB10400
9990 PRINT"YOUR SCORE WAS ";SC:IFSC>75TH
ENPRINT"FANTASTIC WORK, ";N$;"!"
9995 IFSC<25THENPRINT"I'M NOT TOO IMPRES
SED BY YOUR"
9996 IFSC<25THENPRINT"PERFORMANCE..."
9997 GOTO5092
9999 STOP
10000 FORTT=1TO321:NEXT:RETURN
10010 PRINT:PRINT"YOUR MISSION, ";N$;",
IS TO EXPLORE THE"
10020 PRINT"MYSTERIOUS PLANET MARS!":PRI
NT
10030 PRINT"AN ANCIENT CIVILIZATION ONCE
    THRIVED  "
10035 PRINT"HERE.  YOU MUST FIND AS MANY
    OF THE  "
10040 PRINT"TREASURED RELICS FROM THIS L
ONG-DEAD  "
10045 PRINT"CULTURE AS YOU CAN, RETURN T
O YOUR"
10060 PRINT"ROCKET SHIP, AND BLAST OFF F
OR GOOD OLD EARTH.":PRINT
10065 PRINT"BUT WATCH OUT -- SOME ITEMS
YOU WILL  "
10070 PRINT"FIND MAY BE WORTHLESS JUNK T
HAT WILL  "
10075 PRINT"HAVE A NEGATIVE EFFECT ON YO
UR SCORE."
10090 PRINT"SOME OF THE JUNK MIGHT, HOWE
VER, COME  "
10095 PRINT"IN HANDY DURING YOUR EXPLORA
TION OF THE RED PLANET.":PRINT
10110 INPUT"PLEASE PRESS 'RETURN' FOR MO
RE  ";Q$
10120 PRINTCHR$(147):PRINT:PRINT"YOU WIL
L ENCOUNTER A NUMBER OF MARTIAN "
```

```
10125 PRINT"BEASTS AND MONSTERS DURING Y
OUR TRAVELS.";
10130 PRINT"DIFFERENT TYPES OF CREATURES
MUST BE "
10135 PRINT"HANDLED IN DIFFERENT WAYS.
THE "
10140 PRINT"SUCCESSFUL METHODS MAY NOT A
LWAYS BE OBVIOUS."
10160 PRINT"THE BIZARRE NATURAL FORCES O
F MARS WILL"
10165 PRINT"ALSO TEND TO PRESENT OBSTACL
ES.":PRINT:PRINT
10170 PRINT"YOU MAY DEFINE THE CHARACTER
ISTICS OF "
10175 PRINT"YOUR EXPLORER BY SETTING A V
ALUE FROM"
10180 PRINT"1 TO 100% FOR EACH QUALITY (
SUCH AS"
10185 PRINT"STRENGTH, SPEED, ETC.), OR Y
OU CAN LET"
10200 PRINT"THE COMPUTER AUTOMATICALLY D
EFINE YOUR CHARACTER FOR YOU."
10205 PRINT:INPUT"PLEASE PRESS 'RETURN'
FOR MORE " :G$
10210 PRINTCHR$(147):PRINT:PRINT"DURING
THE GAME, EACH COMMAND MAY BE "
10215 PRINT"ONE OR TWO WORDS, SUCH AS --
LOOK,"
10216 PRINT"DROP ROCK, EAT FOOD, WAIT"
10230 PRINT"OR A SINGLE LETTER DIRECTION
AL MOVE "
10235 PRINT"COMMAND: (N=NORTH, S=SOUTH,
E=EAST "
10240 PRINT"W=WEST). COMMANDS CONSISTIN
G OF MORE THAN TWO WORDS, SUCH AS:"
10260 PRINT"'PUT KUFU IN POCKET' ARE NO
T VALID.":PRINT
10265 PRINT"IF YOU HAVE NO IDEA WHAT TO
DO, YOU MAY"
10270 PRINT"USE THE SPECIAL COMMAND 'HEL
P' FOR A "
```

```
10275 PRINT"LIST OF COMMANDS USED IN THE
      GAME."
10290 PRINT"NOT ALL COMMANDS WILL BE ACC
      EPTABLE "
10295 PRINT"UNDER ALL CIRCUMSTANCES! AL
      SO, WHAT "
10300 PRINT"MAY BE HELPFUL AT ONE TIME,
      MAY BE OF"
10310 PRINT"NO PARTICULAR EFFECT AT ANOT
      HER, AND"
10320 PRINT"ACTUALLY HARMFUL AT A THIRD
      TIME. FOR"
10325 PRINT"INSTANCE, EATING IN FRONT OF
      A GRIMPH IS NOT RECOMMENDED.":PRINT
10330 INPUT"PLEASE PRESS 'RETURN' TO DEF
      INE YOUR CHARACTER ";Q$
10335 PRINTCHR$(147)
10340 RETURN
10399 REM * SCORE CALCULATION *
10400 SC=0:FORX=1TO12:IFT(X)=1THENSC=SC+
      10
10410 IFE(X)=R1ANDF(X)=R2THENSC=SC+12
10420 IFJ(X)=1THENSC=SC-2
10430 IFC(X)=R1ANDD(X)=R2THENSC=SC-3.5
10440 NEXT:SC=SC+SV:RETURN
10449 REM * SUPPLIES PRESENT *
10450 IFX=1THENPRINT"SOME FOOD IS HERE."
10460 IFX=2THENPRINT"A BOTTLE OF WATER I
      S HERE."
10470 IFX=3THENPRINT"A KNIFE IS HERE."
10480 IFX=4THENPRINT"A GUN IS HERE."
10490 IFX=5THENPRINT"A LASER IS HERE."
10500 IFX=6THENPRINT"A COIL OF ROPE IS H
      ERE."
10510 IFX=7THENPRINT"AN INFLATABLE RAFT
      IS HERE."
10520 IFX=8THENPRINT"A FLASHLIGHT IS HER
      E."
10530 IFX=9THENPRINT"A METAL PIPE IS HER
      E."
10540 IFX=10THENPRINT"SOME OLD MAGAZINES
      ARE HERE."
```



```
10550 IFX=11THENPRINT"A COMPASS IS HERE.
"
10560 IFX=12THENPRINT"YOUR SPACESUIT IS
HANGING NEATLY ON ITS RACK."
10565 Y=Y+1:RETURN
10569 REM * JUNK PRESENT *
10570 IFX=1THENPRINT"AN OLD SHOE IS HERE
."
10580 IFX=2THENPRINT"A GAUDILY ORNATE RI
NG IS HERE."
10590 IFX=3THENPRINT"A ROCK IS HERE."
10600 IFX=4THENPRINT"A PAIR OF FOSSILIZE
D UNDERSHORTS          IS HERE."
10610 IFX=5THENPRINT"A LARGE CLOT OF DIR
T IS HERE."
10620 IFX=6THENPRINT"AN OLD BONE IS HERE
."
10630 IFX=7THENPRINT"A SHARPENED STICK I
S HERE."
10640 IFX=8THENPRINT"A BADLY CHIPPED URN
  IS HERE."
10650 IFX=9THENPRINT"A PETRIFIED WAD OF
BUBBLE GUM IS HERE."
10660 IFX=10THENPRINT"A COLORFUL FLOWER
IS HERE."
10670 IFX=11THENPRINT"A DEAD BUTTERFLY I
S HERE."
10680 IFX=12THENPRINT"AN INDESCRIBABLE S
LIMY THING IS HERE."
10690 Y=Y+1:RETURN
10699 REM * TREASURES PRESENT *
10700 IFX=1THENPRINT"A DENTED COPPER BOW
L IS HERE."
10710 IFX=2THENPRINT"SOME GOLD COINS ARE
  HERE."
10720 IFX=3THENPRINT"A FOSSILIZED SLIDE
RULE IS HERE."
10730 IFX=4THENPRINT"A STATUE OF A THREE
-ARMED MARTIAN GOD    IS HERE."
```

```
10740 IFX=5THENPRINT"A TARNISHED SILVER
CUP IS HERE."
10750 IFX=6THENPRINT"A GLASS ORB IS HERE
."
10760 IFX=7THENPRINT"A SCROLL INSCRIBED
WITH THE ANCIENT "
10765 IFX=7THENPRINT"MARTIAN LANGUAGE IS
HERE."
10770 IFX=8THENPRINT"SOME GLITTERING STO
NES ARE HERE."
10780 IFX=9THENPRINT"A MYSTERIOUSLY HUMM
ING BOX IS HERE."
10790 IFX=10THENPRINT"A LARGE, POLISHED
SWORD IS HERE."
10800 IFX=11THENPRINT"A BLEACHED SKULL I
S HERE."
10810 IFX=12THENPRINT"A SET OF BLUEPRINT
S FOR AN ANCIENT"
10815 IFX=12THENPRINT"MARTIAN PALACE IS
HERE."
10820 Y=Y+1:RETURN
10849 REM * DEAD MONSTER *
10850 PRINT"THERE IS A DEAD ";
10860 IFQ=-1THENPRINT"SQUEANLEY SERPENT"
;
10870 IFQ=-2THENPRINT"GHOST OF AN ANCIEN
T      MARTIAN HERE";
10880 IFQ=-3THENPRINT"BRINCHLEY BEAST";
10890 IFQ=-4THENPRINT"KUUFU";
10900 IFQ=-5THENPRINT"GRIMPH";
10910 IFQ=-6THENPRINT"PUROFOLEE";
10920 PRINT" HERE. ":X=INT(RND(1)*10+1)
10925 IF(X>7)OR(Q=-3)OR(Q=-5)THENPRINT"THE
SMELL IS HORRENDOUS!"
10930 RETURN
10949 REM * SERPENT *
10950 PRINT"DARN! YOU WERE JUST BITTEN
BY A      SQUEANLEY SERPENT!"
10960 DG=DG-5:RETURN
```

```
10999 REM * GHOST *
11000 PRINT"A GHOST OF AN ANCIENT MARTIA
N SUDDENLY APPEARS BEFORE YOU!"
11010 X=INT(RND(1)*10+1):Y=INT(RND(1)*10
+1):Z=LC(L1,L2)
11020 IFZ<10RZ=2THEN11120
11030 IFZ>7THEN11010
11040 PRINT"'LO,' SAYETH THE GHOST, AT "
:PRINTL1;"":L2;" YOU SHALL FIND A ";
11050 IFZ=1THENPRINT"SQUEANLEY SERPENT"
11060 IFZ=3THENPRINT"BRINCHLEY BEAST"
11070 IFZ=4THENPRINT"KUFU"
11080 IFZ=5THENPRINT"GRIMPH"
11090 IFZ=6THENPRINT"PUROFOLEE"
11100 EX(L1,L2)=Z
11110 PRINT"THE GHOST VANISHES INTO THIN
AIR!":RETURN
11120 PRINT" ", "BOO!":PRINT:GOTO11110
11149 REM * DIA *
11150 X=DX*.8:IFDG>XTHENPRINT"YOU'RE FEE
LING JUST FINE & DANDY!":RETURN
11160 IFDG>150THENPRINT"YOU FEEL VERY GO
OD.":RETURN
11170 IFDG>120THENPRINT"YOU FEEL PRETTY
GOOD.":RETURN
11180 IFDG>105THENPRINT"YOU'RE NOT FEELI
NG TOO BAD, ALL THINGS CONSIDERED."
11185 IFDG>105THENRETURN
11190 IFDG>90THENPRINT"SOME ALKA-SELTZER
MIGHT BE NICE...":RETURN
11200 IFDG>80THENPRINT"YOU'VE HAD BETTER
DAYS.":RETURN
11210 IFDG>70THENPRINT"YOU'RE IN NO SHAP
E TO GO DANCING.":RETURN
11220 IFDG>60THENPRINT"YOU'RE FEELING RA
THER POORLY.":RETURN
11230 IFDG>50THENPRINT"ARE YOUR INSURANC
E PAYMENTS UP TO DATE?":RETURN
11240 IFDG>40THENPRINT"BETTER REHEARSE Y
OUR MOANS AND GROANS.":RETURN
11250 IFDG>30THENPRINT"YOU'RE NOT DOING
WELL AT ALL.":RETURN
```

```
11260 PRINT"IT'S A WONDER YOU CAN STILL  
STAND UP!":RETURN  
11299 REM * INVENTORY *  
11300 PRINT:PRINT"YOU ARE NOW CARRYING -  
-- "  
11310 IFS(1)=1THENPRINT"FOOD"  
11320 IFJ(2)=1THENPRINT"ORNATE RING "  
11330 IFT(3)=1THENPRINT"FOSSILIZED SLIDE  
RULE "  
11340 IFJ(4)=1THENPRINT"FOSSILIZED UNDER  
SHORTS "  
11350 IFS(5)=1THENPRINT"LASER "  
11360 IFJ(6)=1THENPRINT"OLD BONE "  
11370 IFT(7)=1THENPRINT"SCROLL "  
11380 IFJ(8)=1THENPRINT"URN "  
11390 IFS(9)=1THENPRINT"METAL PIPE "  
11400 IFJ(10)=1THENPRINT"FLOWER "  
11410 IFT(11)=1THENPRINT"SKULL "  
11420 IFJ(12)=1THENPRINT"SLIMY THING (?)  
"  
11430 IFS(11)=1THENPRINT"COMPASS "  
11440 IFT(10)=1THENPRINT"LARGE SWORD "  
11450 IFJ(9)=1THENPRINT"PETRIFIED WAD OF  
BUBBLE GUM "  
11460 IFT(8)=1THENPRINT"GLITTERING STONE  
S "  
11470 IFS(7)=1THENPRINT"INFLATABLE RAFT  
"  
11480 IFT(6)=1THENPRINT"GLASS ORB "  
11485 INPUT "PLEASE PRESS RETURN TO CONT  
INUE";Q$  
11490 IFJ(5)=1THENPRINT"CLOT OF DIRT "  
11500 IFT(4)=1THENPRINT"STATUE OF MARTIA  
N GOD "  
11510 IFS(3)=1THENPRINT"KNIFE "  
11520 IFT(2)=1THENPRINT"GOLD COINS "  
11530 IFJ(1)=1THENPRINT"OLD SHOE "  
11540 IFT(1)=1THENPRINT"COPPER BOWL "  
11550 IFS(2)=1THENPRINT"BOTTLE OF WATER  
"  
11560 IFS(2)=2THENPRINT"EMPTY BOTTLE "  
11570 IFJ(3)=1THENPRINT"ROCK "
```

```
11580 IFS(4)=1THENPRINT"GUN  "
11590 IFT(5)=1THENPRINT"SILVER CUP  "
11600 IFS(6)=1THENPRINT"COIL OF ROPE  "
11610 IFJ(7)=1THENPRINT"SHARPENED STICK
"
11620 IFS(8)=1THENPRINT"FLASHLIGHT  "
11630 IFT(9)=1THENPRINT"MYSTERIOUSLY HUM
MING BOX  "
11640 IFS(10)=1THENPRINT"OLD MAGAZINES
"
11650 IFJ(11)=1THENPRINT"BUTTERFLY  "
11660 IFT(12)=1THENPRINT"BLUEPRINTS  "
11670 IFS(12)=1THENPRINT"SPACESUIT  "
11680 Z=0:FOR Y=1 TO 12:Z=Z+S(Y)+J(Y)+T(Y):
NEXT:IF Z=0 THEN PRINT"NOTHING"
11690 PRINT:RETURN
11699 REM * ERROR *
11700 X=INT(RND(1)*8+1):IF X=1 THEN PRINT"S
AY WHAT?"
11710 IF X=2 THEN PRINT"THAT DOES NOT COMPU
TE."
11720 IF X=3 THEN PRINT"DON'T BE SILLY, ";N
$;"!"
11730 IF X=4 THEN PRINT"???!???"
11740 IF X=5 THEN PRINT"I DON'T UNDERSTAND.
"
11750 IF X=6 THEN PRINT"HOW WOULD THAT HELP
HERE?"
11760 IF X=7 THEN PRINT"WHAT ON MARS ARE YO
U BABBLING ABOUT, ";N$;"?"
11770 IF X=8 THEN PRINT"THAT DOESN'T MAKE A
NY SENSE, ";N$;"!"
11780 RETURN
11799 REM * HELP *
11800 PRINT:PRINT"POSSIBLE COMMANDS INCL
UDE --- "
11810 PRINT"BLAST OFF","CLIMB","CRY","DI
A","DRINK",
11820 PRINT"DROP","EAT","FILL","GET","HE
LP","INFLATE",
11830 PRINT"INV","KILL","LOOK","MAP","OP
EN",
```

```
11840 PRINT"PRAY","SCORE","TOUCH","WAIT"
11850 PRINT:PRINT"NOT ALL COMMANDS WILL
BE RECOGNIZED AT ALL TIMES."
11860 PRINT:INPUT"PLEASE PRESS 'RETURN'
TO CONTINUE THE GAME ";Q$
11870 RETURN
11899 REM * RAVINE *
11900 L3=L1:L4=L2:PRINT"YOU JUST FELL IN
TO A DEEP RAVINE!"
11905 DG=DG-INT(RND(1)*DG+1)
11910 FORX=1TO4:Y=INT(RND(1)*11+1):IFT(Y
)=1THEN11980
11920 IFJ(Y)=1THEN11990
11930 IFS(Y)>0THEN12000
11940 NEXT:PRINT:GOSUB10000
11950 PRINT"IT TAKES QUITE A BIT OF EFFO
RT, BUT YOU MANAGE TO CRAWL OUT TO ";
11960 PRINT"THE SOUTH.":PRINT:L1=L1+1:IF
L1>10THENL1=1
11970 Q=LC(L1,L2):EX(L1,L2)=Q:RETURN
11980 T(Y)=0:E(Y)=INT(RND(1)*10+1):F(Y)=
INT(RND(1)*10+1):GOTO11940
11990 J(Y)=0:D(Y)=INT(RND(1)*10+1):F(Y)=
INT(RND(1)*10+1):GOTO11940
12000 S(Y)=0:A(Y)=INT(RND(1)*10+1):B(Y)=
INT(RND(1)*10+1):GOTO11940
12049 REM * MARSQUAKE *
12050 PRINT"THE GROUND BEGINS TO RUMBLE
BENEATH YOUR FEET!":PRINT
12060 L1=INT(RND(1)*5+1)+L1-3:IFL1>10THE
NL1=1
12070 IFL1<1THENL1=10
12080 L3=L1-1:IFL3<1THENL3=10
12090 L2=L2+INT(RND(1)*5+1)-3:IFL2<1THEN
L2=10
12100 IFL2>10THENL2=1
12110 L4=L2:FORX=1TO40
12120 Y=INT(RND(1)*10+1):Z=INT(RND(1)*10
+1):ZZ=LC(Y,Z):IFZZ=20THEN12140
12130 IFZZ>0THENZZ=-ZZ
12140 LC(Y,Z)=ZZ:NEXT
```

```

12150 PRINT " ", "* WHEW! *":PRINT"THE MAR
SQUAKE IS OVER NOW!"
12160 LC(L1,L2)=0:Q=0:EX(L1,L2)=0:RETURN
12199 REM * STORM *
12200 PRINT"YOU ARE CAUGHT IN A WEIRD MA
RTIAN STORM!":PRINT
12210 LC(L1,L2)=0:Q=0:EX(L1,L2)=0
12220 DG=DG-INT(RND(1)*DG/4+1)
12230 FORX=1TO40:Y=INT(RND(1)*10+1):Z=IN
T(RND(1)*10+1)
12235 IFLC(Y,Z)=20THEN12250
12240 LC(Y,Z)=-LC(Y,Z)
12250 NEXT:GOSUB10000
12260 PRINT"THE WEATHER SEEMS TO BE CLEA
RING UP NOW.":PRINT
12270 RETURN
12299 REM * SKY *
12300 PRINT"ODD... THE SKY TURNS A FUNNY
COLOR FOR A FEW MINUTES..."
12310 PRINT:GOSUB10000
12320 FORX=1TO25:Y=INT(RND(1)*10+1):Z=IN
T(RND(1)*10+1)
12330 ZZ=LC(Y,Z):IFZZ=20THEN12360
12335 IF ZZ>12 THEN ZZ=-1
12340 LC(Y,Z)=ZZ+1
12360 NEXT:PRINT" WELL, EVERYTHING SEEMS
TO BE BACK TO NORMAL NOW. ";
12370 GOSUB10000:PRINT"I GUESS...":PRINT
12380 RETURN
12399 REM * MAP *
12400 PRINTCHR$(147):PRINT:PRINT
12410 FORX=1TO10:PRINT" ";;:FOR Y
=1TO10
12415 IFL1=XANDL2=YTHENPRINT"+ ";;:GOTO12
450
12420 Z=EX(X,Y):IFZ<10RZ>9THEN12440
12430 ONZGOTO12500,12510,12520,12530,125
40,12550,12560,12570,12580
12440 IFZ=20THENPRINT"* ";;:GOTO12450
12445 PRINT". ";

```

```
12450 NEXT:PRINT:NEXT:PRINT
12460 PRINT"+= YOU, *= SHIP, S= SQUEANLE
Y SERPENT, "
12465 PRINT"B= BRINCHLEY BEAST, K= KUFU,
G= GRIMPH, "
12470 PRINT"P= PUROFOLEE, R= RIVER, M= M
OUNTAIN, "
12475 PRINT"V= KAVINE":PRINT" PRESS 'R
ETURN' TO CONTINUE GAME ";
12490 INPUTQ$:RETURN
12500 PRINT"S ";;GOTO12450
12510 PRINT". ";;GOTO12450
12520 PRINT"B ";;GOTO12450
12530 PRINT"K ";;GOTO12450
12540 PRINT"G ";;GOTO12450
12550 PRINT"P ";;GOTO12450
12560 PRINT"R ";;GOTO12450
12570 PRINT"M ";;GOTO12450
12580 PRINT"V ";;GOTO12450
12600 IFQ=3THENMM=INT(RND(1)*4+1)
12610 IFQ=4THENMM=INT(RND(1)*6+1)
12620 IFQ=5THENMM=INT(RND(1)*10+1)
12630 IFQ=6THENMM=INT(RND(1)*5+1)
12640 RETURN
```


Chapter 7

Graphics, Sound, and Other Extras

You can create intriguing and creative games by using the described techniques. The programmer's imagination is the only limit to what can happen. However, this still may not be good enough.

All the game techniques presented in the previous chapters are text-oriented. The computer prints out a question; the player chooses and types in his answer; and the computer prints out the results. While this kind of game is fun, it does lack realism. Some players find it very difficult to really get into a purely text-oriented game.

This chapter takes a quick look at a few possible ways you can liven up an adventure game. These extras are not included in any of the complete game programs featured in this book because computer brands vary widely in capabilities and methods. Text-oriented BASIC commands are more universal, although there are some differences. Watch out for unfamiliar commands if you try these programs on anything other than a Commodore 64. Check the user's manual.

However, you should take full advantage of your particular computer's capabilities, if you so choose.

Remember when adding these features to an adventure game that they are extras. A reflex-oriented game, like Space Invaders or PAC-MAN, is built primarily around graphic displays. In an adventure game, on the other hand, graphics enhance the realism and excitement of the game; but they should not be the main point of the game. This is important to remember while programming because both graphics (and many other special features) and basic adventure games each take up quite a bit of memory space. With a

lengthy game program like MARS you may have no room left to add any graphics. You probably should not try to add any unless your system has a very large memory.

It's usually a good idea to write the basic text-oriented game first, then go back and add the extras if you still have enough unused memory space.

Always run a complete sample before checking the memory size to allow for the storage of all the variable values. Strings and arrays can use a lot of memory space. If you fill too much memory space with actual programming, the game may bomb out with an "out of memory" error mid-way through.

Run several samples as you add extras. It's a good idea to leave at least 500 bytes open after a complete sample run just in case you didn't use all of the variables to their maximum limits in the test run. When the available memory size drops below this limit, stop programming. To add anything more will lead to trouble.

GRAPHICS

Graphics, probably the most popular computer extra, are simply computer-generated illustrations.

Programming methods for graphics vary widely from computer to computer.

One of the most common methods, crude but possible on virtually all microcomputers, is to print out selected alphanumeric characters in a specific pattern to form a rough image. Some commonly used characters for this trick are "X", ".", ":", "/", "o", "@", and "*". Figure 7.1 shows a simple example. The drawings made by this technique are crude and awkward; but, in some cases, they may be sufficient. It is a novelty rather than a practical programming tool.

```

////////////////////
:
@ :      *      *      @
:
:      O
:
:      (_____)
:
:      .....
:

```

Figure 7.1 Example of simple, typed graphics.

Most practical computer graphics systems are based on a grid matrix of dots (or individual screen positions) that can be independently turned on or off. For example, the Commodore 64 computer uses a grid matrix of 320 horizontal dots and 200 vertical dots high resolution display. The greater the number of dots in such a grid matrix system, the greater the resolution of the graphics. That is, more dots allow greater detail and smoother appearing simulated arcs and diagonals.

The Commodore 64 has a specific range of memory addressed dedicated to the monitor display screen. You can POKE desired values into the appropriate memory addresses to create an illustration on the screen. For example, in the command POKE X,Y, the X indicates the grid location being specified (an arithmetic expression which can range from 0 to 65535); Y represents the value (can be an integer value of 0 to 255), whether character or color, to be placed in the location indicated by X.

The Commodore 64 displays graphics characters, as well as letters, numbers and symbols. The graphics symbols indicated on the keyboard are accessed by holding down either the SHIFT key or the Commodore Logo key and pressing the key with the desired graphic symbol on the front of it. You can also draw curves or lines at any specified location of the display screen.

Color capabilities are also featured on the Commodore 64. You can change the color of the border and background of the screen by selecting from the 16 options for the border, screen and character color. Special locations in the memory of the Commodore 64 control the colors used. Each screen location may be changed to any of several colors with a single command. For example, you can change the color of the cursor and of the material printed on the screen by holding the CTRL key and pressing a number from 1 to 8. In addition, seven other colors are available. If you hold down the key with the Commodore logo on it and press a number from 1 to 8, a different set of colors is available.

Unfortunately, BASIC is really too awkward and slow for really high-quality graphics and animation. In most cases, you must resort to assembly (or machine) language for graphics routines, even if the main program is written in BASIC.

Most graphics generation systems are tedious and/or tricky to program, at least to some degree. It is usually necessary to go through several sample runs before the graphics illustrations come out exactly the way you want them. You can draw the desired

image on graph paper first. But the picture often looks quite different once you get it on the monitor display screen.

APPLICATIONS FOR GRAPHICS

So how can you incorporate graphics into an adventure game? The programmer's imagination is the ultimate deciding factor. Possibilities are virtually limitless. The following paragraph suggests a few typical applications.

You can display a drawing of the landscape whenever the player arrives at a new location (or commands LOOK or something similar). This landscape view may or may not be animated.

Rather than a verbal description, the items can be directly illustrated. Being told a Kufu is looming over you is never quite the same as actually seeing it.

Some adventure games divide the monitor display screen into two (or more) discrete sections. One displays the standard text of the game, while the other section continuously illustrates the action of the game. In practical programs, due to memory limitations, these continuous illustrations generally tend to be rather crude. Often just simple geometric shapes are used. A triangle might represent the hero, a circle the treasure, and a square the monster. While far from being realistic, these graphics add a great feeling of action to the game. It is especially effective when used along with real-time techniques.

Perhaps you can invoke a special command (SEE, for example) when the player meets a monster. This command would cause the computer to display a static, or simply animated picture of the creature, so the player can judge what he's up against. In the game of MARS, for instance, the Grimp should appear larger and more ferocious than a Brinchley Beast.

A programmer can incorporate some reflex-oriented tasks into an adventure game. In these situations, graphics are combined with what will be called real-time techniques. This is appropriate for battling monsters. The player has to dodge the monster's attack while he aims his own attack. This makes killing a monster more a test of skill than luck, as when the results are determined by randomly generated numbers.

Good graphic routines add considerable appeal to any game. But remember that the game itself should come first. Graphics consume mammoth portions of the computer's memory space. Don't scrimp on variety within the game to allow space for graphics,

because if the game wears thin nobody will bother to play and the graphics won't be seen at all.

SOUND EFFECTS

Sound effects can also add a great deal to the excitement of a good adventure game. For example, a monster roars as it attacks, or screams as it dies. Perhaps the player can hear the hero's labored breath (in MARS) as he crawls out of a ravine.

Musical effects are another possibility. A fanfare could be played when the player vanquishes a monster, or a dirge when the battle turns out the other way.

Many popular microcomputers have limited sound generation capabilities controlled by BASIC. Often this includes only music-like tones, although some models have more varied sound effect possibilities.

With most microcomputers, the programmer can produce a limited range of sounds. The Commodore 64, however, offers a wide range of options if you want to create sound effects to accompany your games. Five different memory locations create sound on the Commodore 64. Each of these memory locations handles a separate aspect of the sound created and provides some of the most amazing music and sound effects available on a personal computer.

Programming sound effects can often be extremely tedious, unless the computer can use specific BASIC commands to generate the desired effect. But it is often worth the trouble to enliven an adventure game.

Several types of circuit boards will accept a computer (digital) output signal and use it to control a variety of sound effects. Perhaps the most versatile and powerful of those now on the market are the AY-3-8910 and AY-3-8912 Programmable Sound Generators. These devices, manufactured by General Instruments, were designed specifically to create computer-controlled sound effects and music.

Each chip generates up to three independent tones and a random noise signal (useful for explosions and percussive, or drum-like, sounds). By modifying and combining these four basic signals, a programmer can generate thousands of complex sounds.

A big advantage of using a device like the AY-3-8910/8912 is that it contains its own internal latches. This means the chip will remember the instructions from the computer until they are over-ridden by new instructions. The computer can perform other tasks

while the Programmable Sound Generator is producing a sound. This can speed up complex programs quite a bit.

A 61-page data and applications manual for the AY-3-8910/8912 Programmable Sound Generator is available from General Instrument dealers for further reference. To repeat, this type of external sound generation device makes computer controlled sound effects much easier.

REAL-TIME INPUTS

Adventure games in this book use straight inputs. That is, the computer will ask a question, then wait patiently for the player to make up his mind and reply. It will wait three weeks, or three years, as long as your electric bills are paid on time. Until the RETURN key is pressed, nothing will happen. The program simply stops until an input is received.

Many people find computer games much more exciting when time limits force them to think fast. They feel more involved in the situation described in the game. This is called real-time play. Action is continuous, rather than stop and start.

Most modern microcomputers have an alternative INPUT command that will not stop the program instantly. On the Commodore 64 this command is GET. Any single character (not whole words) may be entered. The RETURN key does not have to be hit. The computer accepts the input as soon as a key is depressed.

As an example, here is a simple routine that might be used for facing a monster:

```
800 PRINT "YOU ARE FACING A MUQUOLP!"
810 PRINT:PRINT "YOUR CHOICE OF ACTION?"
820 PRINT "A -- ATTACK"
830 PRINT "B -- FLEE"
840 PRINT "C -- CRY":PRINT
850 C$= "":T=1
860 C$=INKEY$:T=T+1:IF T=300 THEN 950
870 IF C$= "" THEN 860
880 REM*COMMAND CHECK ROUTINE*
950 REM*MONSTER ATTACK ROUTINE*
```

Real-time techniques are often used along with graphics and/or sound effects.

For this type of real-time command routine, either print a menu of possible commands (as in the example shown above) and/or make sure that no two commands begin with the same letter.

SAVE GAME

Adventure games tend to be fairly lengthy, and it is not always possible to play an entire game in one sitting. This makes a SAVE GAME feature highly desirable.

At any point throughout the game, the player may choose to SAVE the game. This would probably be programmed along with the rest of the commands. That is, each command check routine would have a new line added, so that it recognizes SAVE as a legitimate command. For instance, in the main command check routine in MARS, we could add the following line:

```
775 IF QX$= "SAV" THEN 12000
```

When the player selects the command SAVE, all current variable values are loaded onto a cassette or a floppy disc.

At the beginning of the game, the player should be asked if he wants to start a new game, or LOAD an old game. If LOAD is selected, the variable values are read back into the computer from the tape or disc. The game will immediately pick up where it left off.

For details on how to SAVE and LOAD variable values, see your computer's manuals. It is unfortunately beyond the scope of this book to discuss the various ways this is done with popular microcomputers.

If you're fairly new to programming, I'd advise you not to bother with SAVE and LOAD routines. They can be rather tricky if you're not 100 percent sure of what you're doing.

SUMMARY

This chapter gives little specific detail, mainly because the techniques for these extra features differ so much from brand to brand. The intent here was simply to suggest a few ideas that might spark up your own imagination when you're working on your own adventure games.

Remember though, these goodies are all frills. You must first create a solid game that will stand up on its own. If you can then

make the game even better by adding graphics, or other extras—great! But fantastic graphics and sound effects are no substitute for a well-thought-out, exciting game.

Chapter 8

Treasure Hunt

One of the most popular and enduring types of adventure fantasies is a hunt for a buried pirate treasure. This idea lends itself very well to the adventure game format.

The TREASURE HUNT game program, in Listing 8.1 is the simplest one in this book. It fits easily into a 16K computer.

In this game you are the captain of a modern-day pirate ship, seeking a treasure that was buried by the crew of the Jolly Roger between 1600 and 1899. (The date is randomly selected in line 280).

The treasure is buried on one of 10 islands in the area of a coral reef. A mermaid lives on the coral reef, and if you are lucky she may tell you that the treasure is not located on a specific island. However, approaching the coral reef is terribly risky, and it may cost you one of your crew members, or even sink your ship and end the game.

As the game begins, you are sailing with a six-man crew. Your crew members are:

ABRAMS
BENNETT
CLANCY
DAWSON
EGBERT
FRED

Any of these characters may be sent ashore to explore an island. The captain (you) can never leave the ship.

While a man is ashore on one of the islands, a simple map is displayed. A typical island map is illustrated in Figure 8.1.

```

X . T .
. . . R Q
. T . .
R . . T .

```

T = Tree
 R = Rock
 Q = Quicksand
 X = Your Man

Figure 8.1 Typical Island Map.

A character may pass through spaces marked as Rocks or Trees, but he will be injured. Stepping into quicksand is immediately fatal. But your crew is well-trained (albeit somewhat stupid). They will obey your orders and walk wherever you tell them to.

Each character has a health rating. Injuries (such as tripping over Rocks, or walking into Trees) decrease this health rating. If the health rating of a character drops to zero (or below), that character will die. If your entire crew is killed off, you lose the game.

Characters can also be killed off by having them walk off the north or south end of an island. The east/west ends loop around continuously to simplify the programming.

A character might also encounter a murderous ghost or a cannibal not displayed on the map. Once you encounter a ghost or cannibal it will not move. You must keep track of where ghosts or cannibals attack your men. The ghost will kill your man outright. He may, or may not, be able to escape the cannibal, depending on his health rating.

Characters encounter bleached human skulls, harmless (albeit disgusting) objects. The only island where more than one skull might appear is the one that contains the treasure. This can be a valuable clue.

The flag of the Jolly Roger is also hidden on one of the islands, but not the one with the treasure. Once you have this flag in your possession, all ghosts will flee.

The game ends when either you locate the treasure (you win), or kill off your entire crew (you lose).

Listing 8.1 Complete TREASURE HUNT Program.

```
10 PRINT CHR$(147);
20 REM * TREASURE HUNT * DELTON T. HORN
   * V1.0
30 DIM A(10,35): DIM B(35)
40 PRINT CHR$(147);: PRINT : PRINT : PRI
NT "THE PIRATES' TREASURE HUNT"
50 PRINT : PRINT : INPUT "      YOUR NA
ME, MATEY ";N$
60 FOR J = 1 TO 10: FOR K = 1 TO 25:Y =
  INT ( RND (1) * 17 + 1)
65 IF Y>7 THEN Y=1
70 PRINT " * ";: IF Y>4 THEN Y=Y-4
80 A(J,K) = Y: NEXT : NEXT
90 PRINT CHR$(147);: PRINT :X = INT ( R
ND (1) * 10 + 1):XZ = X
100 Y = INT ( RND (1) * 25 + 1):A(X,Y)
= 100: FOR Z = 1 TO 3
110 Y = INT ( RND (1) * 25 + 1): IF A(X
,Y) > 1 THEN 110
120 A(X,Y) = 10
130 Y = INT ( RND (1) * 25 + 1): IF A(X
,Y) > 3 THEN 130
140 FOR X = 1 TO 10
150 Y = INT ( RND (1) * 35 + 1): IF A(X
,Y) > 1 THEN 150
160 A(X,Y) = 10
170 Y = INT ( RND (1) * 30 + 1): IF A(X
,Y) > 1 THEN 170
180 A(X,Y) = 30: NEXT
190 X = INT ( RND (1) * 10 + 1): IF X =
  XZ THEN 190
200 Y = INT ( RND (1) * 25 + 1): IF A(X
,Y) > 1 THEN 200
210 A(X,Y) = 40
220 AD = 3:BN = 3:CL = 3:DW = 3:EG = 3:F
R = 3:FL = 0
230 S = INT ( RND (1) * 5 + 1): IF S =
  1 THEN S$ = "SOLEMN ROGER"
```

```
240 IF S = 2 THEN S$ = "LEAKY TUB"
250 IF S = 3 THEN S$ = "SEA DEVIL"
260 IF S = 4 THEN S$ = "SON OF JOLLY ROGER"
270 IF S = 5 THEN S$ = "PIRATE SHIP"
280 YY=(INT(RND(1) * 3 + 1) + 15) * 100:
YR = INT ( RND (1) * 100 + 1) -1 +YY
290 PRINT : PRINT "YOU ARE CAPTAIN ";N$;
","
300 PRINT : PRINT "BLOOD-THIRSTY MASTER
OF THE": PRINT "SS ";S$
310 PRINT:PRINT "YOU ARE SEEKING THE TRE
ASURE BURIED BY THE PIRATE CREW OF "
315 PRINT"THE JOLLY ROGER IN " :YR
330 PRINT:PRINT"YOUR CREW CONSISTS OF ---
- "
340 IF AD>0 THEN PRINT "ABRAMS",
350 IF BN>0 THEN PRINT "BENNETT",
360 IF CL > 0 THEN PRINT "CLANCY",
370 IF DW > 0 THEN PRINT "DAWSON",
380 IF EG > 0 THEN PRINT "EGBERT",
390 IF FR > 0 THEN PRINT "FRED",
395 PRINT : IF FL = 1 THEN PRINT "YOU A
RE CARRYING THE FLAG OF THE"
396 IF FL=1 THEN PRINT"ORIGINAL JOLLY RO
GER"
400 C=AD+BN+CL+DW+EG+FR: IF (C=0) OR (C<
0) THEN 1000
410 PRINT
411 PRINT"THERE ARE 10 ISLANDS IN THE AR
EA."
412 PRINT"(ENTER '11' TO VISIT THE CORAL
REEF). "
413 PRINT"WHICH ISLAND SHALL THE"
414 PRINT S$;:INPUT" VISIT NEXT";I
440 I = INT (I): IF (I > 0) AND (I < 11
) THEN 480
450 IF I = 11 THEN 3000
455 PRINT
460 PRINT "WHAT IN THE NAME OF BLUEBEARD
ARE YOU TALKING ABOUT, CAP'N?!?"
470 GOTO 410
```

```
480 PRINT : PRINT "CREW MEMBER TO GO ASH  
ORE ON ISLAND #";I;" **"  
490 IF AD > 0 THEN PRINT " ", "1 --- ABR  
AMS"  
500 IF BN > 0 THEN PRINT " ", "2 --- BEN  
NETT"  
510 IF CL > 0 THEN PRINT " ", "3 --- CLA  
NCY"  
520 IF DW > 0 THEN PRINT " ", "4 --- DAW  
SON"  
530 IF EG > 0 THEN PRINT " ", "5 --- EGB  
ERT"  
540 IF FR > 0 THEN PRINT " ", "6 --- FRE  
D"  
550 PRINT : PRINT "YOUR ORDER, CAPTAIN "  
;N$;  
560 INPUT C: IF (C = 1) AND (AD > 0) THE  
N 630  
570 IF (C = 2) AND (BN > 0) THEN 630  
580 IF (C = 3) AND (CL > 0) THEN 630  
590 IF (C = 4) AND (DW > 0) THEN 630  
600 IF (C = 5) AND (EG > 0) THEN 630  
610 IF (C = 6) AND (FR > 0) THEN 630  
620 PRINT : PRINT " ", "WHO??";: GOTO 560  
630 CP = 1: FOR X = 1 TO 25: B(X) = A(I, X  
): NEXT  
640 IF CP = 0 THEN PRINT "CAPTAIN ";N$;  
" OF THE "  
645 IF CP=0 THEN PRINT"SS ";S$:GOTO 330  
650 GOSUB 2010  
660 PRINT " IS ASHORE ON ISLAND #";I: PR  
INT  
670 GOSUB 2080  
680 GOSUB 2010  
690 PRINT ": WHICH WAY SHOULD I GO, CA  
P'N";  
700 D$ = "":INPUT D$:D$ = LEFT$ (D$,1)  
710 IF C = 1 THEN AD = AD - .1  
720 IF C = 2 THEN BN = BN - .1  
730 IF C = 3 THEN CL = CL - .1  
740 IF C = 4 THEN DW = DW - .1  
750 IF C = 5 THEN EG = EG - .1
```

```
760 IF C = 6 THEN FR = FR - .1
770 GOSUB 2210
780 IF D$ = "R" THEN CP = 0
790 IF CP = 0 THEN 640
800 IF D$ = "N" THEN 2500
810 IF D$ = "S" THEN 2510
820 IF D$ = "E" THEN 2520
830 IF D$ = "W" THEN 2530
840 GOSUB 2010
850 PRINT ":  WHAT?!?"
860 GOSUB 2000
870 PRINT : PRINT "NORTH, SOUTH, EAST, W
EST, OR RETURN      TO THE SHIP"
880 GOTO 700
909 REM * SPACE CHECK *
910 IF B(CP) = 100 THEN 1240
920 IF B(CP) = 2 THEN 1300
930 IF B(CP) = 3 THEN 1430
940 IF B(CP) = 4 THEN 1460
950 IF B(CP) = 10 THEN 1520
960 IF B(CP) = 20 THEN 1560
970 IF B(CP) = 30 THEN 1620
980 IF B(CP) = 40 THEN 1750
990 GOTO 640
999 STOP
1000 PRINT "NO ONE": GOSUB 2000
1010 PRINT : PRINT : PRINT "WITHOUT A CR
EW TO GUIDE HER, YOUR SHIP "
1015 PRINT "DRIFTS HELPLESSLY ACROSS THE
SEVEN"
1020 PRINT " SEAS THROUGHOUT THE REST OF
ETERNITY,":GOSUB 2000
1040 PRINT:PRINT "YOU LOSE, ";N$:PRINT:
GOSUB 2000
1050 PRINT"INCIDENTALLY, THE TREASURE WA
S BURIED "
1055 PRINT"ON ISLAND #";XZ
1060 PRINT:PRINT:PRINT: INPUT"WOULD YOU
LIKE TO PLAY AGAIN";S$
1070 S$ = LEFT$(S$,1): IF S$ = "Y" THE
N 40
1080 PRINT "OK.  GOOD-BYE."
```

```
1084 REM EXIT ROUTINE PUTS TEXT KQ$ INTO
      KEYBOARD QUEUE STARTING AT 631
1086 F$="MENU"
1088 KQ$="LOF$,8"+CHR$(13)+"RU"+CHR$(13)
1090 FOR I=631 TO 640
1092 POKE I,ASC(MID$(KQ$,I-630,1))
1094 NEXT I
1096 POKE 198,10:REM 10 CHARS IN QUEUE
1098 END
1099 REM * DROWNING *
1100 PRINT : GOSUB 2010
1110 PRINT " : BLUB, BLUB, BLUB ...":
PRINT : PRINT : PRINT
1120 IF C = 1 THEN AD = 0
1130 IF C = 2 THEN BN = 0
1140 IF C = 3 THEN CL = 0
1150 IF C = 4 THEN DW = 0
1160 IF C = 5 THEN EG = 0
1170 IF C = 6 THEN FR = 0
1180 GOSUB 2000: GOSUB 2010
1190 PRINT " WALKED OFF THE EDGE OF THE
ISLAND, AS ORDERED. OF COURSE HE";
1200 PRINT" DROWNS"
1210 GOSUB 2000
1220 INPUT"PLEASE PRESS 'RETURN' ";A$
1225 PRINT CHR$(147)
1230 CP = 0: GOTO 640
1240 PRINT : GOSUB 2010
1250 PRINT " JUST FOUND THE TREASURE!!":
PRINT
1260 GOSUB 2000
1270 PRINT"YOU AND YOUR ENTIRE REMAINING
CREW ARE NOW INDEPENDENTLY WEALTHY!!"
1275 PRINT
1290 GOSUB 2000:GOTO 1060
1300 PRINT : GOSUB 2010
1310 PRINT " JUST RAN INTO A TREE.": PR1
NT
1320 IF C = 1 THEN AD = AD - .125
1330 IF C = 2 THEN BN = BN - .125
1340 IF C = 3 THEN CL = CL - .125
1350 IF C = 4 THEN DW = DW - .125
```



```
1360 IF C = 5 THEN EG = EG - .125
1370 IF C = 6 THEN FR = FR - .125
1380 GOSUB 2010
1390 PRINT ":   OUCH!": PRINT
1400 INPUT "PLEASE PRESS 'RETURN'   ": A$
1405 PRINT CHR$(147)
1410 GOSUB 2210
1420 GOTO 640
1430 PRINT : GOSUB 2010
1440 PRINT " JUST TRIPPED OVER A LARGE ROCK!": PRINT
1450 GOTO 1320
1460 PRINT : GOSUB 2010
1470 PRINT " JUST STEPPED INTO A POOL OF QUICKSAND!": PRINT
1480 GOSUB 2000: PRINT
1490 GOSUB 2010: PRINT ":   ARGH!!!!": PRINT
1500 GOSUB 2280
1510 GOTO 1400
1520 PRINT : GOSUB 2010
1530 PRINT " JUST FOUND A BLEACHED HUMAN SKULL.": PRINT : GOSUB 2000
1540 GOSUB 2010: PRINT ":   ICK!": PRINT
1550 GOTO 1400
1560 IF FL = 1 THEN 1800
1570 PRINT : PRINT "A GHOST KILLS ";
1580 GOSUB 2010: PRINT "!": PRINT
1590 X = INT ( RND (1) * 25 + 1): IF B(X) > 10 THEN 1590
1600 A(1,X) = 20
1610 GOTO 1400
1620 PRINT : GOSUB 2010
1630 PRINT " JUST RAN INTO A HUNGRY CANNIBAL!": PRINT
1640 DC = INT ( RND (1) + 1) * 2: IF C = 1 THEN AD = AD - DC
1650 IF C = 2 THEN BN = BN - DC
1660 IF C = 3 THEN CL = CL - DC
1670 IF C = 4 THEN DW = DW - DC
1680 IF C = 5 THEN EG = EG - DC
```

```
1690 IF C = 6 THEN FR = FR - DC
1700 GOSUB 2210
1710 GOSUB 2000: IF CP = 0 THEN 1740
1720 PRINT "HE JUST BARELY MANAGES TO ES
CAPE WITH HIS LIFE!"
1730 GOTO 1400
1740 PRINT : PRINT "CANNIBAL: BURP!":
PRINT : GOTO 1400
1750 PRINT : GOSUB 2010
1760 PRINT " JUST FOUND THE FLAG OF THE
OLD JOLLY ROGER!": PRINT
1770 FL = 1: GOSUB 2000
1780 B(CP) = 1: A(I,CP) = 1
1790 GOTO 1400
1800 PRINT : PRINT "THE GHOSTS ON THE IS
LAND ALL FLEE FROM "
1805 PRINT "THE FLAG OF THE JOLLY ROGER"
1810 PRINT
1820 FOR X=1 TO 25
1830 IF B(X)=30 THEN B(X)=1
1840 IF A(I,X) = 30 THEN A(I,X) = 1
1850 NEXT : GOTO 1400
2000 FOR TT = 1 TO 345: NEXT : RETURN
2010 IF C = 1 THEN PRINT "ABRAMS";
2020 IF C = 2 THEN PRINT "BENNETT";
2030 IF C = 3 THEN PRINT "CLANCY";
2040 IF C = 4 THEN PRINT "DAWSON";
2050 IF C = 5 THEN PRINT "EGBERT";
2060 IF C = 6 THEN PRINT "FRED";
2065 PRINT "... "
2070 RETURN
2079 REM * MAP *
2080 LC = 1: FOR X = 1 TO 5: PRINT "
": FOR Y = 1 TO 5
2090 IF CP = LC THEN 2170
2100 IF B(LC) = 2 THEN 2180
2110 IF B(LC) = 3 THEN 2190
2120 IF B(LC) = 4 THEN 2200
2130 PRINT ". ";
2140 LC = LC + 1: NEXT Y: PRINT : NEXT X
2150 PRINT : PRINT "T=TREE, R=ROCK, Q=QU
ICKSAND, X=YOUR MAN": PRINT
```

```
2160 RETURN
2170 PRINT "X ";; GOTO 2140
2180 PRINT "T ";; GOTO 2140
2190 PRINT "R ";; GOTO 2140
2200 PRINT "Q ";; GOTO 2140
2210 IF AD < 0 THEN PRINT "ABRAMS IS DE
AD":CP = 0:AD = 0: RETURN
2220 IF BN < 0 THEN PRINT "BENNETT IS D
EAD":CP = 0:BN = 0: RETURN
2230 IF CL < 0 THEN PRINT "CLANCY IS DE
AD":CP = 0:CL = 0: RETURN
2240 IF DW < 0 THEN PRINT "DAWSON IS DE
AD":CP = 0:DW = 0: RETURN
2250 IF EG < 0 THEN PRINT "EGBERT IS DE
AD":CP = 0:EG = 0: RETURN
2260 IF FR < 0 THEN PRINT "FRED IS DEAD
":CP = 0:FR = 0: RETURN
2270 RETURN
2279 REM * KILL CHARACTER *
2280 IF C = 1 THEN AD = - 1
2290 IF C = 2 THEN BN = - 1
2300 IF C = 3 THEN CL = - 1
2310 IF C = 4 THEN DW = - 1
2320 IF C = 5 THEN EG = - 1
2330 IF C = 6 THEN FR = - 1
2340 RETURN
2499 REM * MOVE *
2500 CP = CP - 5: GOTO 2540
2510 CP = CP + 5: GOTO 2540
2520 CP = CP + 1: GOTO 2540
2530 CP = CP - 1
2540 IF (CP < 1) OR (CP > 25) THEN 1100
2550 GOTO 910
3000 GOSUB 2000:SS = INT ( RND (1) * 15
+ 1): IF SS = 13 THEN 3100
3010 G = INT ( RND (1) * 8 + 1): IF (G
= 1) AND (AD < 0.01) THEN 3200
3020 IF (G = 2) AND (BN < 0.01) THEN 320
0
3030 IF (G = 3) AND (CL < 0.01) THEN 320
0
```

```
3040 IF (G = 4) AND (DW < 0.01) THEN 320
0
3050 IF (G = 5) AND (EG < 0.01) THEN 320
0
3060 IF (G = 6) AND (FR < 0.01) THEN 320
0
3070 IF G > 6 THEN 3200
3080 PRINT : PRINT "IN APPROACHING THE D
ANGEROUS REEF,":C = G: GOSUB 2010
3085 PRINT " IS KILLED!"
3090 GOSUB 2280: GOTO 3200
3100 PRINT:PRINT "THE SS ";S$;" SINKS!":
PRINT
3110 GOTO 1040
3200 PRINT : PRINT "A BEAUTIFUL MERMAID,
WHO LIVES ON THE "
3205 PRINT"CORAL REEF, TELLS YOU THE TRE
ASURE"
3206 PRINT"IS NOT BURIED ON ISLAND #":
3210 H=INT(RND(1)*10+1):IF H=XZ THEN 321
0
3220 PRINT H:PRINT
3230 INPUT"PLEASE PRESS 'RETURN' " :A$
3235 PRINT CHR$(147)
3240 GOSUB 2210
3250 CP=0:GOTO 640
```

PROGRAMMING

The TREASURE HUNT program is broken down into routines in Table 8.1. Table 8.2 lists all of the variables used in this program. Figures 8.2A, 8.2B, 8.2C, and 8.2D are flowcharts, and Figure 8.3 is a summary.

The game features 10 small islands. A map of each island is stored in an individual array. The 10 arrays are actually a single two-dimensional array. The first coordinate identifies the island. The second coordinate is the location on that island. For example, array location A(5,7) is position 7 on island 5.

A second, single-dimension array is used to represent the island currently being visited. This array (B(35)) is used to display the island maps.

Table 8.1 Routines and Subroutines Used in the TREASURE HUNT Program.

Routines

10-50	initialize
60-220	preset variables
230-270	name ship
280-320	year treasure was buried
330-400	crew listing
410-470	island selection
630	set current island
640-680	island display
690-880	input command
910-990	check contents of current location
1000-1050	entire crew dead
1060-1090	new game?
1100-1230	crew member drowns
1240-1290	treasure found
1300-1420	crew member runs into a tree
1440-1450	crew member trips over rock
1460-1510	crew member steps into quicksand
1520-1550	crew member finds skull
1560-1610	ghost attacks crew member
1620-1740	cannibal attacks crew member
1750-1790	crew member finds flag
1800-1850	ghosts flee from flag

Subroutines

2000	time delay
2010-2070	print crew member name
2080-2200	display island map
2210-2270	crew member dead announcement
2280-2340	crew member dies
2500-2550	calculate move
3000-3250	visit coral reef

Each island is arranged in a 5-X-5 format, as shown in Figure 8.1. There are 25 spaces on each island. The arrays are dimensioned to 35 so that not all of the obstacles will appear on every island. Obstacles planted at array locations 26 through 35 are functionally invisible. This is somewhat wasteful of memory space, but that should not be a problem for this game.

Table 8.2 Variables Used in the TREASURE HUNT Program.

AD	Abrams' health rating
BN	Bennett's health rating
C	crew member to go ashore
CL	Clancy's health rating
CP	crew member's position on island
DC	effect of cannibal attack
DW	Dawson's health rating
EG	Egbert's health rating
FL	flag found?
FR	Fred's health rating
G	crew member killed by coral reef
H	mermaid's island choice
I	island to visit
J,K	preset counting loops
S	ship name select
SS	does ship sink at coral reef?
X,Y,Z	misc.
YR,YY	year of burial

string variables

D\$	directional command
N\$	Captain's (player's) name
S\$	ship name/play again?

arrays

A(10,35)	main island maps
B(35)	current island map

After the string space is cleared (line 10) and the arrays dimensioned (line 30), the player's name is requested, and stored as N\$.

Next, the island maps are preset. The values used to represent each object on the islands are summarized in Table 8.3.

In lines 60 through 80, each active island space (up to array location 25) is assigned a value from 1 to 4. A 1 is a blank location, A 2 is a tree, A 3 is a rock, and A 4 represents a pool of quicksand. The value for each space is assigned randomly. A random number from 1 to 17 is selected, and then manipulated to a value of 1, 2, 3, or 4. Table 8.3 shows the stored value for each possible value of Y. Notice that a 1 is the most likely value, and a 4 is the least likely.

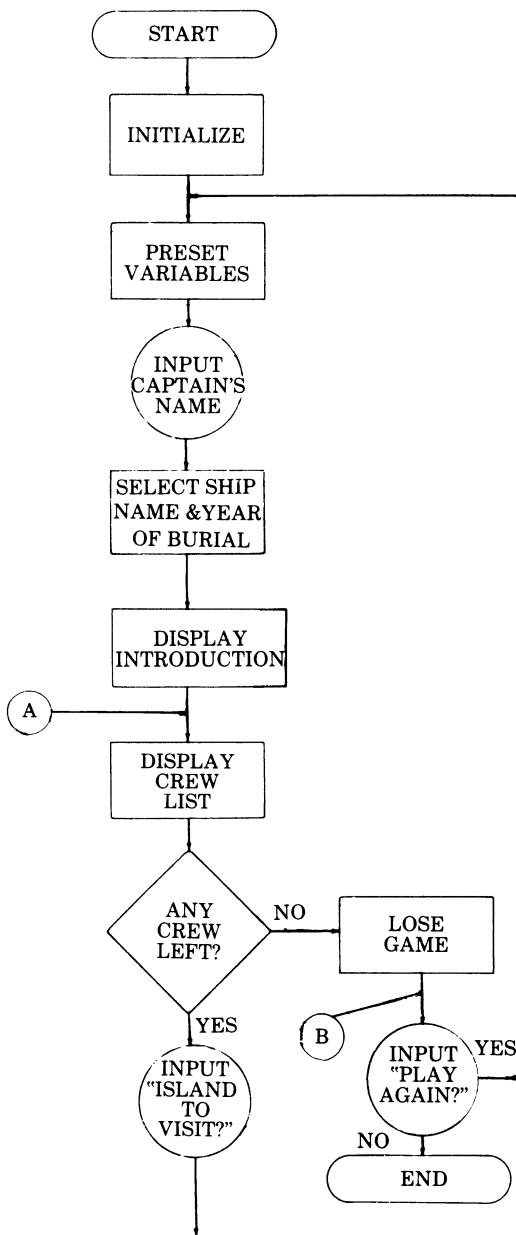


Figure 8.2A Flow-chart for TREASURE HUNT Program.

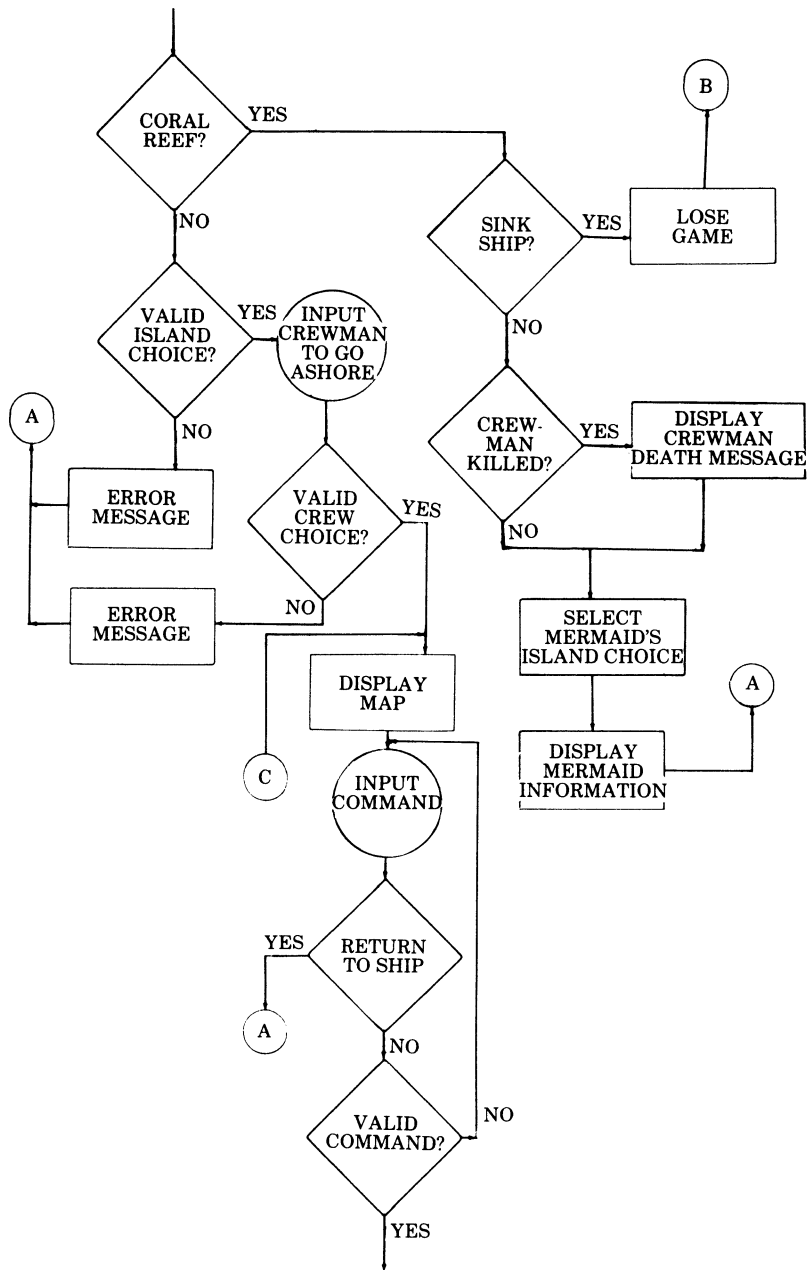


Figure 8.2B Flow-chart for TREASURE HUNT Program.

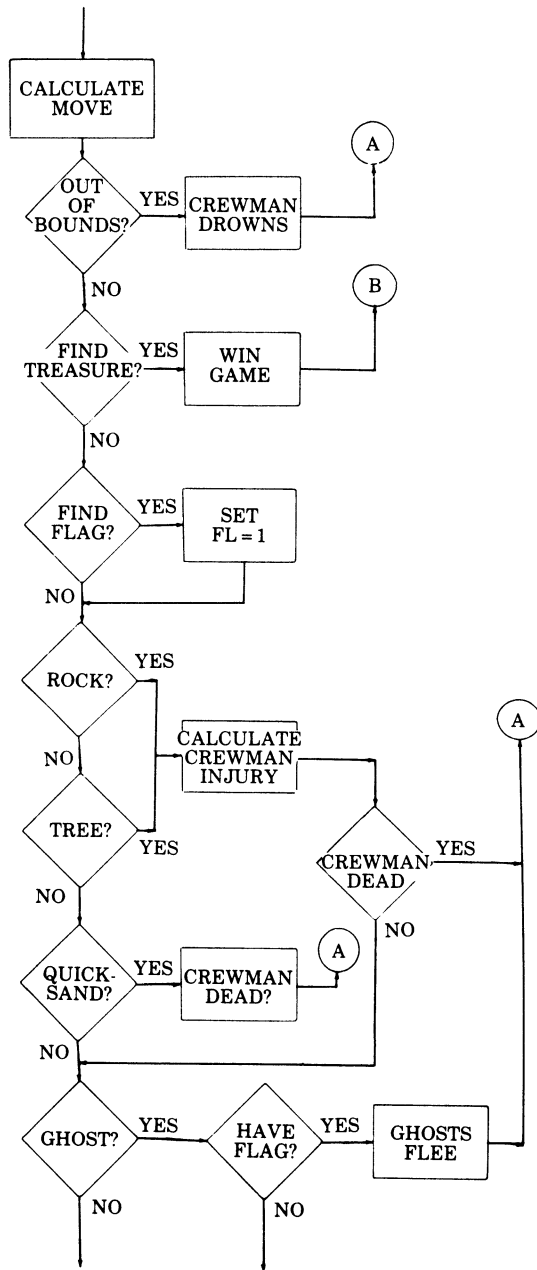


Figure 8.2C Flow-chart for TREASURE HUNT Program.

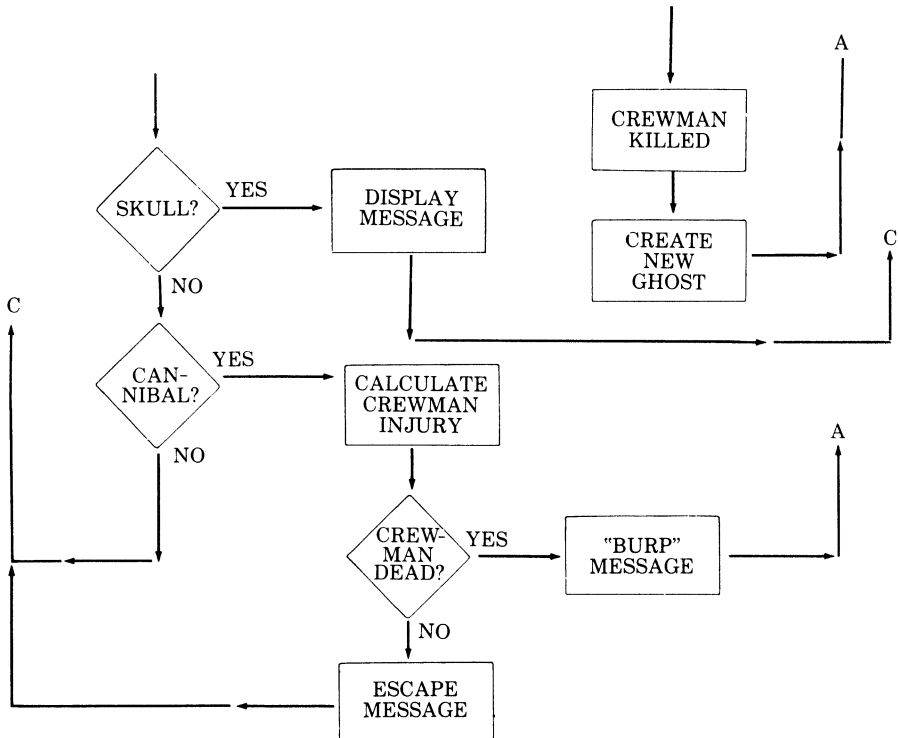


Figure 8.2D Flow-chart for TREASURE HUNT Program.

Game location — Aboard a pirate ship in an area with ten islands and a coral reef

Hero/player character — Captain of the ship

Mission/Goal — To find the treasure hidden on one of the islands

Additional characters — The Captain's crew (Abrams, Bennett, Clancy, Dawson, Egbert, and Fred), Mermaid on coral reef (source of information), ghosts (obstacles), cannibals (obstacles)

Other obstacles — quicksand pits, approaching the coral reef, drowning, rocks (trip over), trees (walk into)

Figure 8.3 Summary of the TREASURE HUNT Game.

Table 8.3 Object Values for the TREASURE HUNT Game.

1	blank space
2	tree
3	large rock
4	pool of quicksand
10	bleached human skull
20	ghost
30	cannibal
40	flag of the "Jolly Roger"
100	the treasure

A random value from 1 to 10 is assigned to X in line 90. This same value is also stored as XZ. This variable indicates which island holds the treasure. The treasure (value of 100) is planted randomly in one of the 25 active locations on the selected island in line 100.

Three skulls are randomly located on the treasure island (value of 10) in lines 100 through 120. When a location is randomly selected, if that location is not blank (the value is greater than 1), a new location is selected.

A ghost (value of 20) is also placed on the island with the treasure in lines 130 and 135. If the random location contains the treasure, a skull, or a pool of quicksand (value greater than 3), a new location is selected.

In lines 140 through 180 a skull (10) and a cannibal (30) are placed on each of the 10 islands. Notice that this brings the total number of skulls on the island with the treasure (XZ) up to 4. These skulls may be placed at any array location up to 35. Skulls placed at locations 26 through 35 will not show up during the game. Similarly, cannibals may be placed anywhere from location 1 to 30, but only those in locations 1 to 25 will appear during the game. This way a skull or cannibal may or may not be found on each of the islands.

Finally, an island is selected for the flag (line 190). If the treasure island (XZ) is selected, a new island will be chosen. The flag may only be placed in one of the 25 active island locations. Its value is 40 (lines 200 and 210).

The preset routine is completed in line 220, where each of the six crew character health ratings (AD, BN, CL, DW, EG, and FR) are assigned an initial value of 3. The variable FL is preset to 0 to indicate that the flag has not been found.

Lines 230 through 270 randomly select one of five names for the player's pirate ship. You may substitute other names of your own creation. Alternatively, you could assign a permanent name that will be the same for every game, or substitute an INPUT statement so the player can select any name he likes for the ship.

A year is selected in line 280. A century is selected as YY. This variable may take on a value of 1600, 1700, or 1800. YR adds from 0 to 99 to this, making the year anything from 1600 to 1899. In the opening message (lines 290 through 320), this value is displayed as the year in which the treasure was buried. All of this is just for show and has no effect on the game itself. Touches like this can make the game more interesting.

Because this game is quite simple and self-prompting, no separate instruction routine is included in the program. If you want to include an instruction subroutine, you should call it at this point.

Lines 330 through 400 begin the actual game. The crew is displayed. Notice that the health rating of each crew member is checked, and the name displayed only if the crew member is alive (health rating greater than 0). For example:

```
340 IF AD > 0 THEN PRINT "ABRAMS",
```

Line 395 determines if the flag of the Jolly Roger has been found, and prints an appropriate message if it has.

In line 400 all of the crew health ratings are combined. If their combined value is 0 or less, the entire crew has been killed off, and the program jumps to the lose game routine at line 1000.

On the first round of the game, all six crew member names should be displayed.

THE PLAY

The player is prompted to select an island (1 through 10) to visit, or the coral reef (11) in lines 410 through 430. The player's input is checked for validity in lines 440 and 450. If an incorrect value has been entered, an error message (line 460) is displayed, and the player is asked for a new input. The player is not penalized for an incorrect entry.

Assume the player has entered 11 to visit the mermaid on the coral reef. The program jumps to line 3000. There is one chance in 15 ($6\frac{2}{3}$ percent) that the ship will sink (lines 3100 and 3110). Of course, this means the player loses the game.

If the ship is not sunk, a random value from 1 to 8 (G) is selected. If this value is 7 or 8, nothing special happens, and the program jumps ahead to line 3200. Each of the other six values represent each of the crewmen. If the selected crewman is not already dead, he will be killed.

In either case, the player selects a random number from 1 to 10 (H). If H is equal to XZ (the island with the treasure) a new value is selected for H. Then a beautiful mermaid, who lives on the coral reef, tells you the treasure is NOT buried on island (H).

This information may, or may not be useful. It could allow you to eliminate that island from your search. On the other hand, the mermaid may tell you about an island you have already visited.

Visiting the coral reef is dangerous. There is a good chance that it will kill one of your crew. This means you should consult the mermaid sparingly. In any adventure game, frequent use of such advisors should be discouraged to prevent the game from getting too easy. If, for instance, there was no risk in visiting the coral reef, the player could keep going back until the mermaid identifies nine islands without the treasure. Then he visits the one remaining island to win the game unfairly.

On most moves, the player will select one island to visit. He is asked which crew member will be sent ashore. Only living crew members are acceptable (see lines 480 through 620). The number representing the selected crewman is stored as C.

The selected island map is put into temporary array B(x) (line 630), and the variable CP is set to equal 1. This variable is used to indicate the crewman's position on the island. Any visit to an island will always begin at array location 1. Any obstacle at this point (tree, rock, quicksand, ghost, or cannibal) will be ignored. If, however, the man is moved and then is sent back to position 1, the obstacle will behave in its normal manner.

A subroutine beginning at line 2080 displays the island map (as illustrated in Figure 8.1). The crewman then asks the Captain (the player) for his orders (lines 680 through 700). Only the first letter of the player's input (D\$) is relevant. There are five acceptable commands:

- N – move one space north (up)
- S – move one space south (down)
- E – move one space east (left)
- W – move one space west (right)
- R – return to the ship

The character may be returned to the ship from any position on the island.

An incorrect command will produce an error message, and a list of recognized commands (lines 840 through 880).

Notice that each command costs the crewman 0.1 from his health rating. This is true of both valid and invalid commands. Even if there were no obstacles, each character can only survive a total of 30 commands.

The contents of each location are checked in lines 910 through 990.

If the location has a value of 100, the treasure has been located and the program jumps to line 1240 for the win game routine.

ENCOUNTERING OBSTACLES

If the map location has a value of 2, the player is informed that his crewman just ran into a tree (lines 1300 and 1310). That character's health rating is reduced by 0.125. Basically, the same thing happens when the location has a value of 3, except the displayed message states that the crewman tripped over a large rock (lines 1430 through 1450).

A pool of quicksand is represented by a location value of 4. If you order your character to step into quicksand, he will be killed (lines 1460 through 1510).

Trees, rocks, and quicksand pools are displayed on each island map, so you can avoid them whenever possible. Occasionally, it may be necessary to pass a tree or a rock to get to another part of the island; but remember that your crew member will be injured. It is never good strategy to send a man into quicksand.

Other obstacle objects are not displayed on the island maps; but once found, they will not move (except the flag, and ghosts fleeing from the flag).

A location value of 10 represents a bleached human skull. A message is displayed that expresses the crew member's disgust (lines 1520 through 1550). Ordinarily, finding a skull has no direct effect on the game. However, if you find more than one skull on a single island, you can assume the treasure is hidden on that island.

If a ghost is encountered (location value is 20), the value FL (flag position) will be checked (line 1560). If FL equals 1, indicating the flag has been found, the program is diverted to line 1800, where all of the ghosts are removed from the appropriate island arrays. A message stating that all of the ghosts on the island flee from the flag of the Jolly Roger will be displayed.

Ghosts will only be encountered on the island with the treasure.

If FL equals 0, the flag has not been found. The ghost will immediately kill the crewman (lines 1570 and 1580), and his ghost is placed somewhere on the island (lines 1590 and 1600) for future haunting.

The game starts out with only one ghost, but you might end up with up to six haunting the treasure island (the original ghost and five dead crewmen). If the character is killed by anything other than a ghost, he will not return to haunt you.

The final obstacle is the cannibal (location value of 30). A random value of up to 2 (not necessarily an integer) is subtracted from the character's health rating (lines 1640 through 1700). If this value is enough to kill the character, the cannibal has a nice meal (see line 1740). Otherwise, the crewman manages to escape (line 1720), but he will be weakened by the encounter.

The location value might also be 40. This represents the flag of the Jolly Roger. The location value is set back to 1, and flag possession variable FL is set to 1 (lines 1750 through 1790).

POSSIBLE VARIATIONS

For an easier game, you could add more crew members, and/or increase their initial health ratings (line 220). For example, you might set the initial values at 4 or 5.

You can create a harder game by lowering the initial health ratings to 2, or 2.5. I do not recommend starting with health ratings lower than this because the game might become impossible to win.

For more variety, you could assign different health ratings for each of the characters. In this case, some of the men would die more readily than others. You could set the initial health ratings randomly, like this:

```
220 AD = (INT (RND (1)*8+1))/2 :
    BN = (INT (RND (1)*8+1))/2 :
    CL = (INT (RND (1)*8+1))/2 :
    DW = (INT (RND (1)*8+1))/2 :
    EG = (INT (RND (1)*8+1))/2 :
    FR = (INT (RND (1)*8+1))/2 :
    FL = 0
```

This line would give each character an initial health rating from 0.5 to 4.0 (in steps of 0.5). Some players may find this variation too frustrating.

Table 8.4 Odds for Blank Spaces, Trees, Rocks, and Quicksand Pools in the TREASURE HUNT Game.

(lines 60-80)

Y	adjusted value	object
1	1	blank
2	2	TREE
3	3	ROCK
4	4	QUICKSAND
5	1	blank
6	2	TREE
7	3	ROCK
8	1	blank
9	1	blank
10	1	blank
11	1	blank
12	1	blank
13	1	blank
14	1	blank
15	1	blank
16	1	blank
17	1	blank
ODDS		
1	blank	70.5%
2	TREE	12%
3	ROCK	12%
4	QUICKSAND	5.5%

SUMMARY

The TREASURE HUNT program is one of the simplest forms for an adventure game, but it is still plenty of fun to play. Included here, it demonstrates that you don't have to be an expert in fancy programming tricks, or spend months writing hundreds of lines, to create an enjoyable and worthwhile game program.

Chapter 9

The Golden Flute

Fantasy and fairy tales provide another profitable source for adventure game ideas. Elves, ogres, giants and gremlins all make fine characters in an adventure game. Adult fantasy has enjoyed a new popularity since *The Lord of the Rings* was published, and such concepts are perfect for the adventure gamester.

This chapter will program a fantasy-oriented adventure game called THE GOLDEN FLUTE. The story goes as follows.

Love is brought into the world when Brombiran, the Lord of the Woodland Elves, plays his magical Golden Flute. Unfortunately, the evil gremlin king, Terak, has stolen the Golden Flute. Now you must join the creatures of the Woodlands in a noble quest to recover the Golden Flute of Love.

PURPOSE

In this game, the player must guide a Magic Chariot through the playing area in search of Terak's secret hideaway. Once the Golden Flute is in his possession, the player must return it safely to the Woodlands. The object of the game is to reach this goal in as few moves as possible. Naturally, along the way the player will encounter numerous monsters and obstacles. He may also find magical weapons and gold pieces.

The complete program for THE GOLDEN FLUTE is Listing 9.1. Table 9.1 breaks the program down into its component routines. Figures 9.1A, 9.1B, 9.1C, and 9.1D provide you with a flowchart. And Table 9.2 lists the possibilities.

Listing 9.1 Complete THE GOLDEN FLUTE Program.

```
10 REM * THE GOLDEN FLUTE * DELTON T. HO
RN * V1.0
19 REM * SET UP *
20 CLR
30 DIM A(100): DIM B(100): DIM C(9)
40 PRINT CHR$(147):PRINT : PRINT : PRINT
  "THE GOLDEN FLUTE"
50 PRINT : PRINT " ", "BY DELTON T. HORN"
  : PRINT : PRINT
55 PW = 0:GW = 0
60 INPUT "YOUR NAME ";N$
70 BR = 1:DG = 1:GI = 1:RU = 1:SJ = 1:JS
  = 1:AL = 1:PM = 1
80 AX = 0:LC = 1:M = 1:SW = 0:FL = 0:MD
  = 0:GL = 0:MZ = 0
90 FOR X = 1 TO 100:A(X) = 0:B(X) = 0: N
EXT
95 FOR X = 1 TO 9:C(X) = 1: NEXT
100 PRINT : PRINT "SETTING VARIABLES": P
RINT : PRINT
109 REM * WOODS *
110 A(1) = 1:A(2) = 1:A(11) = 1:A(12) =
  1
119 REM * ORACLE *
120 A(33) = 2
129 REM * PITS*
130 X=INT(RND(1)*7+1)+2:Y= INT(RND (1) *
  4 + 1) + 4:X = X + Y * 10:Y = X + 1
140 A(X) = 3:A(Y) = 3:X = X + 10:Y = Y +
  1:A(X) = 3:A(Y) = 3
149 REM * WALL *
150 Q = 0
160 Q=Q+1:X=(INT(RND(1)*4+1)+3)*10+INT(R
ND (1) * 3 + 1) + 3:Y = X + 5:Z = 0
170 IF Q > 7 THEN 220
180 FOR V = X TO Y: IF A(V) > 0 THEN Z =
  1
190 IF A(V + 10) THEN Z = 1
200 NEXT : IF Z = 1 THEN 160
```

```
210 FOR V = X TO Y:A(V) = 13:A(V + 10) =  
14: NEXT  
220 FOR X = 1 TO 100:B(X) = A(X): PRINT  
"%";: NEXT : PRINT  
229 REM * HIDDEN ITEMS *  
230 X = INT ( RND (1) * 100 + 1)  
232 IF B(X) > 0 THEN 230  
235 B(X) = 29  
239 REM * TERA & GUARDS *  
240 X=(INT(RND(1)*4+1)+4)*10+INT(RND(1)*  
7+1)+2: IF B(X) > 0 THEN 240  
250 B(X) = 4:Z = 9:Y = X - 1: GOSUB 5010  
260 Y = X + 10: GOSUB 5010  
270 Y = X - 10: GOSUB 5010  
280 Y = X + 1: GOSUB 5010  
290 Y = Y + 10: GOSUB 5010  
300 Y = Y - 20: GOSUB 5010  
309 REM * GARGOYLES *  
310 U=INT(RND(1)*10+1)+2:FOR X=1TOU:Y=IN  
T(RND(1)*100+1):GOSUB 5010: NEXT  
320 Y = INT ( RND (1) * 70 + 1) + 30: P  
RINT "*";: IF B(Y) > 0 THEN 320  
330 B(Y) = 5  
340 Y = INT ( RND (1) * 100 + 1): IF B(  
Y) > 0 THEN 340  
350 B(Y) = 21: FOR X = 1 TO 3  
360 Y = INT ( RND (1) * 100 + 1): IF B(  
Y) > 0 THEN 360  
370 B(Y) = 22: NEXT  
380 Y = INT ( RND (1) * 50 + 1): IF B(Y  
) > 0 THEN 380  
390 B(Y) = 23  
400 Y = INT ( RND (1) * 50 + 1): IF B(Y  
) > 0 THEN 400  
410 B(Y) = 24  
420 Y = INT ( RND (1) * 90 + 1): IF B(Y  
) > 0 THEN 420  
430 B(Y) = 25  
440 Y = INT ( RND (1) * 80 + 1) + 15: I  
F B(Y) > 0 THEN 440  
450 B(Y) = 26: FOR X = 1 TO 5
```

```
460 Y = INT ( RND (1) * 80 + 1) + 15: I
F B(Y) > 0 THEN 460
470 B(Y) = 27: NEXT
480 Y = INT ( RND (1) * 100 + 1): IF B(
Y) > 0 THEN 480
490 B(Y) = 26
500 Z = 6: FOR X = 1 TO 10:Y = INT ( RN
D (1) * 80 + 1) + 20: GOSUB 5010: NEXT
510 Z = 8: FOR X = 1 TO 10:Y = INT ( RN
D (1) * 100 + 1): GOSUB 5010: NEXT
520 Y = INT ( RND (1) * 100 + 1): IF B(
Y) > 0 THEN 520
530 B(Y)=7:Z=10:FOR X=1 TO 10:Y=INT(RND(
1) * 95 + 1) + 5: GOSUB 5010: NEXT
540 Y = INT ( RND (1) * 100 + 1): IF B(
Y) > 0 THEN 540
550 B(Y) = 11:QQ = 0
560 F = INT ( RND (1) * 100 + 1): IF B(
F) > 0 THEN 560
570 B(F) = 15
580 Y = INT ( RND (1) * 55 + 1) + 45: I
F B(Y) > 0 THEN 580
590 B(Y) = 17:BZ = 0
599 REM * INTRO *
600 PRINT CHR$(147) : PRINT : PRINT "
    GREETINGS, ";N$: PRINT
610 PRINT "YOU HAVE JOINED THE MAGICAL C
REATURES "
615 PRINT "OF THE WOODLANDS IN THEIR GLO
RIOUS "
620 PRINT "QUEST TO RECOVER THE GOLDEN F
LUTE OF "
625 PRINT "LOVE FROM THE CLUTCHES OF THE
EVIL "
626 PRINT "GREMLIN, TERAk, AND RETURN IT
TO THE"
640 PRINT "WOODLANDS. AT ALL TIMES WATC
H OUT FOR "
645 PRINT "DRAGONS, SIRENS, GOBLINS, GAR
GOYLES, "
```

```
646 PRINT "AND THE DREADED HOPELESS PITS  
!": PRINT  
660 PRINT "CORAMBLE, THE GREAT ORACLE OF  
PURLICON "  
665 PRINT "MOUNTAIN MAY BE OF ASSISTANCE  
." : PRINT  
680 Q$="": INPUT "DO YOU WANT DIRECTIONS"  
; Q$: Q$ = LEFT$(Q$,1)  
690 IF Q$ = "Y" THEN PRINT CHR$(147): GO  
SUB 5030  
695 LC = 1: A(LC) = 12  
699 REM * CHARIOT CONTENTS *  
700 PRINT : GOSUB 5000  
710 PRINT " THE MAGIC CHARIOT IS NOW C  
ARRYING --"  
720 PRINT N$; " THE HUMAN"  
730 IF BR = 1 THEN PRINT "BROMBIRAN THE  
ELF"  
740 IF DG = 1 THEN PRINT "DAGGLETTE THE  
ELF"  
750 IF G1 = 1 THEN PRINT "GROMPHLUR THE  
ELF"  
760 IF RU = 1 THEN PRINT "RULF THE SATY  
R"  
770 IF SJ = 1 THEN PRINT "SEJJAN THE FA  
IRY"  
780 IF JS = 1 THEN PRINT "JESSAN THE FA  
IRY"  
790 IF AL = 1 THEN PRINT "ALLEGRECIA TH  
E QUEEN OF THE SPRITES"  
800 IF PM = 1 THEN PRINT "PRINCESS MELV  
A"  
810 IF FL = 1 THEN PRINT "THE GOLDEN FL  
UTE"  
820 IF MO = 1 THEN PRINT "A MAGIC ORB"  
830 IF SW = 1 THEN PRINT "A MAGIC SWORD  
"  
840 IF BZ = 1 THEN PRINT "A MAGIC BAZOO  
KA"  
850 IF GL < 1 THEN 880
```

```
860 PRINT GL;" PIECE";: IF GL > 1 THEN
PRINT "S";
870 PRINT " OF GOLD"
880 IF MZ = 1 THEN PRINT "A MAGIC ZITHE
R"
890 PRINT : INPUT "PLEASE PRESS 'RETURN'
";Q$
895 PRINT CHR$(147)
899 REM * MAIN PLAY *
900 ZZ = LC: PRINT : PRINT "YOU ARE AT L
OCATION ";LC,"MOVE #";M
910 Q$="":INPUT "YOUR MOVE? (ENTER 'K
' FOR KEY) ";Q$:Q$ = LEFT$(Q$,1)
920 IF Q$ = "K" THEN 4000
925 IF Q$ = "M" THEN 7000
930 IF Q$ = "B" AND BZ = 1 THEN 4030
935 M = M + 1:A(LC) = B(LC)
940 IF Q$ = "D" THEN 1000
950 IF Q$ = "U" THEN 1190
960 IF Q$ = "R" THEN 1200
970 IF Q$ = "L" THEN 1220
980 PRINT " ", "INVALID MOVE!"
990 GOTO 910
1000 LC = LC + 10: IF LC > 100 THEN 1020
1010 GOTO 4250
1020 PRINT : PRINT "THE MAGIC CHARIOT HA
S LEFT THE "
1025 PRINT "BOUNDARIES OF THE MAGIC KING
DOM": PRINT
1030 GOSUB 5000
1040 PRINT "WITHOUT MAGIC TO HOLD IT UP,
THE MAGIC CHARIOT CRASHES!"; PRINT
1050 GOSUB 5000:M = M - 1: PRINT "YOU LO
SE THIS TIME."
1060 PRINT "IT TOOK YOU ";M;" MOVES."
1070 IF PW = 0 OR PW = M THEN 1100
1080 PRINT "PREVIOUS WIN RECORD WAS ";PW
1100 PRINT
1110 GW = GW + 1: PRINT "THIS WAS GAME #
";GW
1120 PRINT " ",: INPUT "PLAY AGAIN";Q$:Q
$ = LEFT$(Q$,1)
```

```
1130 IF Q$ = "Y" THEN 8000
1140 KQ$="LOF$,B"+CHR$(13)+"RU"+CHR$(13)
:F$="MENU"
1141 FOR I=631 TO 640
1142 POKE I,ASC(MID$(KQ$,I-630,1))
1143 NEXT I
1144 POKE 198,10:END::REM 10 CHARS IN QUE
EUE
1150 M = M - 1: PRINT "YOU WIN THIS TIME
": PRINT
1155 PRINT "IT TOOK YOU ";M;" MOVES!"
1160 IF PW = 0 THEN 1175
1170 PRINT "PREVIOUS BEST SCORE WAS ";PW
: IF PW < M THEN 1100
1175 PW = M: GOTO 1100
1190 LC = LC - 10: IF LC < 1 THEN 1020
1195 GOTO 4250
1200 LC=LC+1:X=LC/10:Y=(X-INT (X)) * 10:
IF (Y = 0) OR (Y > 1.5) THEN 4250
1210 GOTO 1020
1220 LC = LC - 1:X = LC / 10:Y = (X - I
NT (X)) * 10: IF Y = 0 THEN 1020
1230 GOTO 4250
4000 PRINT "U=UP, D= DOWN, R= RIGHT, L=
LEFT,": PRINT "M= MAP, ";
4010 IF BZ > 0 THEN PRINT "B = FIRE BAZ
OOKA, "
4020 PRINT "K= KEY": GOTO 910
4030 PRINT "BAZOOKA LOADED --- AIM?"
4035 INPUT "(ENTER 'X' TO DISARM) ";Q$
4040 Q$ = LEFT$(Q$,1): IF Q$ = "X" THE
N 900
4050 IF Q$ = "U" THEN 4100
4060 IF Q$ = "D" THEN 4160
4070 IF Q$ = "R" THEN 4190
4080 IF Q$ = "L" THEN 4210
4090 GOTO 4030
4100 BU = 10000:BS = 1:BT = - 10
4110 PRINT " ", "KER-";: FOR X = 1 TO 55:
NEXT : PRINT "POW!!": PRINT :X = LC
4120 X = X + BT: IF X < BS THEN 900
4130 IF X > BU THEN 900
```



```
4140 IF (X < 1) OR (X > 100) THEN 900
4150 A(X) = 20:B(X) = 20: GOTO 4120
4160 BU = 10000:BS = 0:BT = 10
4170 GOTO 4110
4180 PRINT " ", "PHFFFT!": PRINT : GOTO 900
4190 G = INT (LC / 10): IF G = LC / 10 THEN 4180
4200 BS = 0:BU = G * 10 + 10:BT = 1: GOTO 4110
4210 G = INT (LC / 10):H = LC / 10 - G: IF H = .1 THEN 4180
4220 BU = 10000:BS = G * 10 + 1:BT = -1: GOTO 4110
4250 PRINT : PRINT : PRINT :A(LC) = 12
4260 XX = B(LC)
4270 IF (LC = 1) OR (LC = 2) OR (LC = 11) OR (LC = 12) THEN 10000
4280 IF LC = 33 THEN GOSUB 5120
4290 IF (LC > 21 AND LC < 25) OR (LC > 41 AND LC < 45) THEN GOSUB 5300
4300 IF LC = 32 OR LC = 34 THEN GOSUB 5300
4310 IF XX = 4 THEN GOSUB 5310
4320 P = INT ( RND (1) * 150 + 1): IF (P > 148) AND (FL = 0) THEN 10030
4330 IF XX = 3 THEN 10080
4340 IF XX = 5 THEN 10200
4350 X = LC - 1: GOSUB 5580
4360 X = LC + 1: GOSUB 5580
4370 X = LC - 10: GOSUB 5580
4380 X = X - 1: GOSUB 5580
4390 X = X + 2: GOSUB 5580
4400 X = LC + 10: GOSUB 5580
4410 X = LC - 1: GOSUB 5580
4420 X = LC + 2: GOSUB 5580
4430 IF XX = 6 THEN 10240
4440 IF XX = 7 THEN GOSUB 5600
4450 IF XX = 8 THEN GOSUB 5610
4460 IF XX = 9 THEN 10250
4470 IF XX = 10 THEN 10260
4480 IF XX = 11 THEN GOSUB 5630
```

```
4490 IF XX = 13 OR XX = 14 THEN 11700
4500 IF XX = 15 THEN 11780
4510 IF XX = 17 THEN GOSUB 5640
4520 IF XX = 20 THEN PRINT "THIS AREA I
S A SMOULDERING RUIN."
4525 IF XX = 21 THEN 10270
4527 IF XX = 22 THEN 10280
4530 IF XX = 23 THEN GOSUB 5660
4540 IF XX = 24 THEN GOSUB 5670
4550 IF XX = 25 THEN GOSUB 5750
4560 IF XX = 26 THEN GOSUB 6000
4570 IF XX = 27 THEN 11800
4580 IF XX = 28 THEN GOSUB 6010
4590 IF XX = 29 THEN GOSUB 6050
4900 GOTO 700
4999 STOP
5000 FOR TT = 1 TO 234: NEXT : RETURN
5009 REM * BLANK SPACE CHECK *
5010 IF B(Y) = 0 THEN B(Y) = 2
5020 PRINT "#";: RETURN
5029 REM * INSTRUCTIONS *
5030 PRINT "EACH MOVE MAY BE UP, DOWN, R
IGHT, OR "
5035 PRINT "LEFT. ONLY THE FIRST LETTER
IS NEEDED."
5040 PRINT "NEVER LEAVE THE MAP BOUNDARI
ES! IF YOU"
5045 PRINT "WANT TO SEE A MAP OF THE MAG
IC KINGDOM,"
5050 PRINT "ENTER 'M' AS YOUR MOVE. THIS
WILL NOT"
5060 PRINT "INCREMENT YOUR MOVE COUNTER.
IF YOU HAVE"
5065 PRINT "THE MAGIC BAZOOKA, YOU MAY E
NTER 'B' TO"
5070 PRINT "FIRE. TO HAVE TERAk'S LOCAT
ION "
5075 PRINT "DISPLAYED ON THE MAP, YOU MU
ST VISIT"
5090 PRINT "THE ORACLE ON PURLICON MOUNT
AIN. IF YOU"
```

```
5095 PRINT "ARE CARRYING A MAGIC ORB HE  
MAY EVEN "  
5096 PRINT "TELL YOU MORE. OR HE MAY NO  
T. THE "  
5097 PRINT "OBJECT OF THE GAME IS TO RET  
URN THE"  
5110 PRINT "GOLDEN FLUTE TO THE WOODLAND  
S IN THE FEWEST MOVES."  
5115 PRINT : PRINT "PLEASE PRESS 'RETURN  
' ";:INPUT Q$:PRINT CHR$(147):RETURN  
5120 IF XX = 20 THEN 5290  
5130 PRINT "CORAMBLE, THE GREAT ORACLE,  
REVEALS "  
5135 PRINT "THAT TERAk'S LAIR IS AT LOCA  
TION #";  
5150 FOR Y = 1 TO 100: IF B(Y) = 4 THEN  
5170  
5160 NEXT : IF MD = 1 THEN 5180  
5165 RETURN  
5170 PRINT Y:A(Y) = 4: GOTO 5160  
5180 IF AX = 3 THEN RETURN  
5190 AX=AX+1:A=INT(RND(1)*6+1)+4:IF (A=5)  
OR (A = 6)OR (A=9)OR (A=10)THEN 5210  
5200 RETURN  
5210 PRINT "HE ALSO REVEALS THE LOCATION  
": PRINT "OF ALL ";  
5220 IF A = 5 THEN PRINT "SIRENS."  
5230 IF A = 6 THEN PRINT "DRAGONS."  
5240 IF A = 9 THEN PRINT "GARGOYLES."  
5250 IF A = 10 THEN PRINT "GOBLINS."  
5260 PRINT "THEY WILL BE SHOWN ON YOUR N  
EXT MAP."  
5270 FOR X = 1 TO 100: IF B(X) = A THEN  
A(X) = B(X)  
5280 NEXT : RETURN  
5290 PRINT "THE GREAT ORACLE'S BODY LIES  
SMOKING IN THE CORNER.": RETURN  
5300 PRINT "YOU ARE AT THE FOOT OF PURLI  
CON MOUNTAIN.": RETURN  
5310 PRINT "YOU HAVE INFILTRATED TERAk'S  
LAIR!": GOSUB 5000
```

```
5320 IF FL = 1 THEN 5360
5330 FL = 1: PRINT "YOU RECOVER THE GOLD
EN FLUTE!"
5340 IF MO > 0 THEN 5540
5350 RETURN
5360 D = INT ( RND (1) * 8 + 1): IF D =
  1 AND BR = 0 THEN 5530
5370 IF D = 2 AND DG = 0 THEN 5530
5380 IF D = 3 AND G1 = 0 THEN 5530
5390 IF D = 4 AND RU = 0 THEN 5530
5400 IF D = 5 AND SJ = 0 THEN 5530
5410 IF D = 6 AND JS = 0 THEN 5530
5420 IF D = 7 AND AL = 0 THEN 5530
5430 IF D = 8 AND PM = 0 THEN 5530
5440 IF D = 1 THEN PRINT "BROMBIRAN";:B
R = 0
5450 IF D = 2 THEN PRINT "DAGGLETTE";:D
G = 0
5460 IF D = 3 THEN PRINT "GROMPHLUR";:G
1 = 0
5470 IF D = 4 THEN PRINT "RULF";:RU = 0
5480 IF D = 5 THEN PRINT "SEJJAN";:SJ =
  0
5490 IF D = 6 THEN PRINT "JESSAN";:JS =
  0
5500 IF D = 7 THEN PRINT "ALLEGRECIA";:
AL = 0
5510 IF D = 8 THEN PRINT "PRINCESS MELV
A";:PM = 0
5520 PRINT " IS DEAD.": PRINT
5530 RETURN
5540 A = INT ( RND (1) * 3 + 1): IF A =
  1 THEN 5360
5550 IF A = 3 THEN 5570
5560 RETURN
5570 PRINT "THE MAGIC ORB IS DESTROYED!"
:MO = 0: RETURN
5580 IF (X < 1) OR (X > 100) THEN RETUR
N
5590 IF B(X)=5 THEN PRINT "AN EERIE SWEE
T SINGING IS HEARD IN THE DISTANCE."
```

```
5595 RETURN
5600 PRINT : PRINT "YOU JUST FOUND A MAG
IC SWORD!":B(LC) = 0:SW = 1: RETURN
5610 X =INT(RND(1)*199+1) + 1: PRINT "YO
U JUST FOUND ";X;" PIECES OF GOLD!"
5620 B(LC) = 0:GL = GL + X: PRINT : RETU
RN
5630 PRINT "YOU JUST FOUND A MAGIC ORB!"
: PRINT :MO = 1:B(LC) = 0: RETURN
5640 B(LC) = 0: PRINT "YOU JUST FOUND A
";: GOSUB 5000
5650 PRINT "***** MAGIC BAZOOKA *****":B
Z = 1: RETURN
5660 G$ = "KLUFFOOT": GOTO 5680
5670 G$ = "FRIEK"
5680 IF FL = 0 THEN RETURN
5690 PRINT G$; ", THE GARGOYLE SLAVE OF T
ERAK,"
5700 IF SW = 1 THEN 5720
5710 PRINT " STEALS THE GOLDEN FLUTE AGA
IN!":FL = 0: RETURN
5720 PRINT " ATTEMPTS TO STEAL THE GOLDE
N FLUTE AGAIN!": PRINT
5730 PRINT "BUT THE MAGIC SWORD KILLS ";
G$; "!": PRINT
5740 B(LC) = 0: RETURN
5750 PRINT "YOU ARE IN THE ENCHANTED FOR
EST!": PRINT
5760 ZX = BR + DG + G1 + RU + SJ + JS +
AL + PM
5770 IF ZX > 5 THEN RETURN
5780 R = INT ( RND (1) * 10 + 1): IF R
> 8 THEN RETURN
5790 IF (R = 1) AND (BR = 0) THEN 5880
5800 IF (R = 2) AND (DG = 0) THEN 5890
5810 IF (R = 3) AND (G1 = 0) THEN 5900
5820 IF (R = 4) AND (RU = 0) THEN 5910
5830 IF (R = 5) AND (SJ = 0) THEN 5920
5840 IF (R = 6) AND (JS = 0) THEN 5930
5850 IF (R = 7) AND (AL = 0) THEN 5940
5860 IF (R = 8) AND (PM = 0) THEN 5950
5870 RETURN
```

```
5880 PRINT "BROMBIRAN";:BR = 1: GOTO 597
0
5890 PRINT "DAGGLETTE";:DG = 1: GOTO 597
0
5900 PRINT "GROMPHLUR";:G1 = 1: GOTO 597
0
5910 PRINT "RULF";:RU = 1: GOTO 5970
5920 PRINT "SEJJAN";:SJ = 1: GOTO 5970
5930 PRINT "JESSAN";:JS = 1: GOTO 5970
5940 PRINT "ALLEGRECIA";:AL = 1: GOTO 59
70
5950 PRINT "PRINCESS MELVA";:PM = 1
5970 C(R) = 1.5:B(LC) = 0: PRINT " IS MA
GICALLY RESTORED TO LIFE!"
5980 PRINT : RETURN
6000 PRINT "YOU JUST FOUND A MAGIC ZITHE
R!": PRINT :MZ = 1:B(LC) = 0: RETURN
6010 PRINT"THE MAGIC CHARIOT JUST FLEW O
VER AN      ENCHANTED LAND MINE!":PRINT
6020 GOSUB 5000: FOR X = 1 TO 50:XY = I
NT ( RND (1) * 24 + 1)
6025 XZ = INT ( RND (1) * 40 + 1)
6026 PRINT"";:NEXT PY
6029 PRINT "*";
6030 U = XZ * X: NEXT : PRINT : PRINT
6040 GOTO 5360
6050 IF GL < 2 THEN RETURN
6060 PRINT "A WICKED WITCH STEALS ALL OF
YOUR GOLD COINS!": PRINT
6070 GL = 0: PRINT : PRINT "      HE
E HEE HEE!": PRINT : PRINT
6080 RETURN
6999 REM * MAP *
7000 PRINT CHR$(147):PRINT :PRINT :Z = 1
: FOR X = 1 TO 10: PRINT "      ";
7010 FOR Y = 1 TO 10:V = A(Z)
7020 IF (V=0)OR(V=7)OR(V=8)OR(V=11)OR(V>1
5ANDV<20)OR(V>22ANDV<27)THENPRINT". ";
7030 IF V=1 THEN PRINT "W ";
7040 IF V=2 THEN PRINT "M ";
7050 IF V=3 THEN PRINT "P ";
7060 IF V = 4 THEN PRINT "T ";
```

```
7070 IF V = 5 THEN PRINT "S ";
7080 IF V = 6 THEN PRINT "D ";
7090 IF V = 9 THEN PRINT "G ";
7100 IF V = 10 THEN PRINT "O ";
7110 IF V = 12 THEN PRINT "C ";
7120 IF (V = 13) OR (V = 14) THEN PRINT
    "X ";
7130 IF V = 15 THEN PRINT "F ";
7140 IF V = 20 THEN PRINT "* ";
7150 IF V = 21 THEN PRINT "B ";
7160 IF V = 22 THEN PRINT "A ";
7170 IF V = 27 THEN PRINT "R ";
7180 IF V = 28 THEN PRINT "! ";
7190 IF V = 29 THEN PRINT "H ";
7200 Z = Z + 1: NEXT Y: PRINT : NEXT X
7205 PRINT
7210 PRINT "C= THE MAGIC CHARIOT (YOU),
D= DRAGON "
7215 PRINT "A= DWARF, F= FOG, G= GARGOYL
E, O= GOBLIN";
7216 PRINT "M= PURLICON MOUNTAIN, P= THE
HOPELESS"
7230 PRINT "PITS, R= ROCK, S= SIRENS, B=
MAGIC "
7235 PRINT "SPARROW, T= TERAk'S LAIR, W=
WOODLANDS"
7236 PRINT "H= WITCH, != LAND MINE, *= S
MOULDERING RUIN, X= WALL"
7250 GOTO 900
8000 FOR X = 1 TO 100: A(X) = 0: B(X) = 0:
    PRINT "&";: NEXT
8010 GOTO 70
10000 PRINT "YOU ARE IN THE WOODLANDS, H
OME OF THE ELVES.": PRINT
10010 IF FL = 1 THEN PRINT "YOU HAVE RE
COVERED THE GOLDEN FLUTE!": GOTO 1150
10020 GOTO 4280
10030 PRINT "TERAK FEARS YOUR APPROACH A
ND MOVES HIS LAIR!": PRINT
10040 FOR X = 1 TO 100: IF B(X) = 4 THEN
    B(X) = 0: A(X) = 0
10050 NEXT
```

```
10060 X = INT ( RND (1) * 100 + 1): IF
B(X) > 1 THEN 10060
10070 B(X) = 4: GOTO 4310
10080 PRINT "THE MAGIC CHARIOT IS MIRED
IN THE      HOPELESS PITS!": PRINT
10090 IF MD = 1 THEN 10130
10100 A=INT(RND(1)*5+1): IF A > 3 THEN
PRINT "YOU ARE DOOMED!": GOTO 1050
10110 IF A > 3 THEN GOSUB 5360
10120 PRINT "YOU MANAGE TO GET THE MAGIC
CHARIOT FREE OF THE MUCK.": GOTO 700
10130 INPUT "DO YOU RUB YOUR MAGIC ORB";
Q$:Q$ = LEFT$ (Q$,1)
10140 IF Q$ = "Y" THEN 10160
10150 GOTO 10100
10160 GOSUB 5000: PRINT CHR$(147): PRINT
: PRINT : PRINT : PRINT
10170 PRINT " ", "*** POOF ***": PRINT :
PRINT : PRINT
10180 LC = INT ( RND (1) * 100 + 1)
10185 PRINT "THE MAGIC CHARIOT IS MAGICA
LLY      "
10186 PRINT"TRANSPORTED TO LOCATION #";L
C
10190 A(ZZ) = B(ZZ):A(LC) = 12: GOTO 700
10200 PRINT : PRINT "THE SWEET SONG OF T
HE SIRENS MESMERIZES YOU!": PRINT
10210 GOSUB 5000
10220 PRINT "THE MAGIC CHARIOT CRASHES!"
: PRINT : GOSUB 5000
10230 GOTO 1050
10240 PRINT "A DRAGON ";:FO = 1: GOTO 10
300
10250 PRINT "A GARGOYLE ";:FO = 2: GOTO
10300
10260 PRINT "A GOBLIN ";:FO = 3: GOTO 10
300
10270 PRINT "A MAGIC SPARROW ";:FO = 4:
GOTO 10300
10280 PRINT "AN ANCIENT DWARF ";:GM = I
NT (RND (1) * 150 + 1):FO = 5
```



```
10300 AM=INT(RND (1) * 4 + 1): PRINT "IS
      IN YOUR PATH!": PRINT : GOSUB 5000
10310 A(LC) = B(LC)
10320 PRINT " ", "POSSIBLE ACTIONS": PRIN
T
10330 PRINT "1 --- MOVE THE MAGIC CHARIO
T"
10340 PRINT "2 --- THROW SOME GOLD COINS
      OVERBOARD"
10350 PRINT "3 --- HAND TO HAND COMBAT"
10360 IF MD = 1 THEN PRINT "4 ---- RUB M
      AGIC ORB"
10370 IF SW = 1 THEN PRINT "5 ---- UNSHE
      ATH MAGIC SWORD"
10380 IF MZ = 1 THEN PRINT "6 --- PLAY
      MAGIC ZITHER"
10390 PRINT : INPUT "                YOUR CH
      OICE";P
10400 IF P = 1 THEN 10490
10410 IF P = 2 THEN 10660
10420 IF P = 3 THEN 11000
10430 IF (MD = 1) AND (P = 4) THEN 10160
10440 IF (SW = 1) AND (P = 5) THEN 11500
10450 IF (MZ = 1) AND (P = 6) THEN 11630
10460 PRINT "INVALID SELECTION!": PRINT
      : GOTO 10320
10490 IF FO > 3 THEN 910
10500 PRINT : INPUT "DIRECTION";M$:M$ =
      LEFT$ (M$,1)
10520 IF M$ = "U" THEN 10570
10530 IF M$ = "D" THEN 10580
10540 IF M$ = "R" THEN 10590
10550 IF M$ = "L" THEN 10600
10560 GOTO 10500
10570 IF AM = 1 THEN 1190
10575 GOTO 10610
10580 IF AM = 2 THEN 1000
10585 GOTO 10610
10590 IF AM = 3 THEN 1200
10595 GOTO 10610
10600 IF AM = 4 THEN 1220
```

```
10610 PRINT : PRINT "    THE ";: IF FO =  
1 THEN PRINT "DRAGON";  
10620 IF FO = 2 THEN PRINT "GARGOYLE";  
10630 IF FO = 3 THEN PRINT "GOBLIN";  
10640 PRINT " WILL NOT LET YOU GO THAT W  
AY!"  
10650 GOTO 10320  
10660 PRINT : INPUT "HOW MUCH GOLD DO YO  
U TOSS OVERBOARD";H  
10670 H = ABS (H):H = INT (H): IF H =  
0 THEN 10660  
10680 IF H > GL THEN 10995  
10690 GL = GL - H  
10700 IF FO = 1 THEN 10900  
10710 IF FO = 2 THEN 10910  
10720 IF FO = 3 THEN 10940  
10730 IF FO = 4 THEN 10980  
10740 PRINT "THE DWARF THANKS YOU VERY P  
OLITELY "  
10750 GM = GM - H: IF GM > 1 THEN 10320  
10760 PRINT "AND HE REVEALS THE LOCATION  
OF ALL "  
10770 DF = INT ( RND (1) * 4 + 1): IF D  
F = 1 THEN 10840  
10780 IF DF = 2 THEN 10860  
10790 IF DF = 3 THEN 10880  
10800 PRINT "SIRENS": FOR X = 1 TO 100:  
IF B(X) = 5 THEN A(X) = 5  
10810 NEXT  
10820 PRINT "THIS INFORMATION WILL BE DI  
SPLAYED      ON YOUR NEXT MAP."  
10830 A(LC) = 12: GOTO 700  
10840 PRINT "GOBLINS": FOR X = 1 TO 100:  
IF B(X) = 10 THEN A(X) = 10  
10850 NEXT : GOTO 10820  
10860 PRINT "GARGOYLES": FOR X = 1 TO 10  
0: IF B(X) = 9 THEN A(X) = 9  
10870 NEXT : GOTO 10820  
10880 PRINT "DRAGONS": FOR X = 1 TO 100:  
IF B(X) = 6 THEN A(X) = 6  
10890 NEXT : GOTO 10820
```

```

10900 PRINT "DRAGONS HAVE NO INTEREST IN
      GOLD.": PRINT : GOTO 10320
10910 PRINT "THE GARGOYLE PUTS THE GOLD
      INTO ITS SACK ";
10920 I = INT ( RND (1) * 200 + 1): IF
      H < I THEN PRINT : GOTO 10320
10930 PRINT "AND LEAVES":B(LC) = 0:A(LC)
      = 12: GOTO 700
10940 I = INT ( RND (1) * 200 + 1): PRI
      NT "THE GOBLIN EATS THE GOLD GREEDILY"
10950 IF H < I THEN PRINT : GOTO 10320
10960 PRINT "AND DIES OF": PRINT "TERMIN
      AL INDIGESTION.": PRINT
10970 B(LC) = 0:A(LC) = 12: GOTO 700
10980 PRINT "WHAT WOULD A BIRD WANT WITH
      ";H: PRINT "PIECES OF GOLD?": PRINT
10990 GOTO 10320
10995 PRINT "YOU DO NOT HAVE ";H;" PIECE
      S OF GOLD!": GOSUB 5360: GOTO 10320
11000 PRINT : PRINT " ", "YOUR CHAMPION?"
11010 IF BR = 1 THEN PRINT "1 ---- BROMB
      IRAN"
11020 IF DG = 1 THEN PRINT "2 --- DAGGL
      ETTE"
11030 IF G1 = 1 THEN PRINT "3 --- GROMP
      HLUR"
11040 IF RU = 1 THEN PRINT "4 --- RULF"
11050 IF SJ = 1 THEN PRINT "5 ---- SEJJA
      N"
11060 IF JS = 1 THEN PRINT "6 --- JESSA
      N"
11070 IF AL = 1 THEN PRINT "7 ---- ALLEG
      RECIA"
11080 IF PM = 1 THEN PRINT "8 ---- PRINC
      ESS MELVA"
11090 PRINT "9 ---- ";N$
11100 INPUT "          YOUR CHOICE";CH
11110 IF CH = 1 AND BR = 1 THEN 11210
11120 IF CH = 2 AND DG = 1 THEN 11220
11130 IF CH = 3 AND G1 = 1 THEN 11230
11140 IF CH = 4 AND RU = 1 THEN 11240
11150 IF CH = 5 AND SJ = 1 THEN 11250

```

```
11160 IF CH = 6 AND JS = 1 THEN 11260
11170 IF CH = 7 AND AL = 1 THEN 11270
11180 IF CH = 8 AND PM = 1 THEN 11280
11190 IF CH = 9 THEN 11290
11200 PRINT " ", "WHO???": PRINT : GOTO 1
1010
11210 I = 50: GOTO 11300
11220 I = 55: GOTO 11300
11230 I = 45: GOTO 11300
11240 I = 85: GOTO 11300
11250 I = 25: GOTO 11300
11260 I = 25: GOTO 11300
11270 I = 50: GOTO 11300
11280 I = 40: GOTO 11300
11290 I = 75
11300 H = I * C(CH):C(CH) = C(CH) - .1:
GOSUB 5000
11310 PRINT : PRINT :RS = INT ( RND (1)
* 100 + 1)
11320 IF FO = 4 THEN 11390
11330 IF FO = 5 THEN 11490
11340 IF RS > H THEN 11390
11350 PRINT "THE ";: IF FO = 1 THEN PRI
NT "DRAGON";
11360 IF FO = 2 THEN PRINT "GARGOYLE";
11370 IF FO = 3 THEN PRINT "GOBLIN";
11380 PRINT " IS SLAIN!": PRINT :B(LC) =
O:A(LC) = 12: GOTO 700
11390 IF CH = 1 THEN PRINT "BROMBIRAN";
:BR = 0
11400 IF CH = 2 THEN PRINT "DAGGLETTE";
:DG = 0
11410 IF CH = 3 THEN PRINT "GROMPHLUR";
:G1 = 0
11420 IF CH = 4 THEN PRINT "RULF";:RU =
O
11430 IF CH = 5 THEN PRINT "SEJJAN";:SJ
= 0
11440 IF CH = 6 THEN PRINT "JESSAN";:JS
= 0
11450 IF CH = 7 THEN PRINT "ALLEGRECIA"
;:AL = 0
```

```
11460 IF CH = 8 THEN PRINT "PRINCESS ME
LVA";:PM = 0
11470 IF CH = 9 THEN PRINT N$;" IS SLAI
N!": PRINT "GAME OVER!": GOTO 1050
11480 C(CH) = 0: PRINT " IS SLAIN!": PRI
NT : GOTO 10320
11490 PRINT "THE DWARF IS SLAIN!":B(LC)
= 0:A(LC) = 12: GOTO 700
11500 Q = INT ( RND (1) * 100 + 1): IF
FO = 1 THEN 11550
11510 IF FO = 2 THEN 11600
11520 IF FO = 3 THEN 11350
11530 IF FO = 4 THEN 11390
11540 IF FO = 5 THEN 11490
11550 IF Q < 65 THEN 11350
11560 PRINT "THIS DRAGON IS IMMUNE TO YO
UR SWORD!": PRINT : GOSUB 5000
11570 PRINT "IT ATTACKS THE MAGIC CHARIO
T!": PRINT : GOSUB 5000
11580 GOSUB 5360: IF Q < 80 THEN 10320
11590 PRINT "YOUR MAGIC SWORD IS DESTROY
ED!": PRINT :SW = 0: GOTO 10320
11600 IF Q < 50 THEN 11350
11610 PRINT "THE GARGOYLE DRAWS ITS OWN
MAGIC SWORD!": PRINT : GOSUB 5000
11620 GOTO 11580
11630 IF FO = 1 THEN 11350
11640 IF FO = 3 THEN 11670
11650 PRINT "NO ONE IS PARTICULARLY IMPR
ESSED WITH YOUR TALENT.": PRINT
11660 GOTO 10320
11670 Q = INT ( RND (1) * 10 + 1): IF Q
> 5 THEN 11350
11680 GOSUB 5360: GOTO 10320
11699 REM * WALL *
11700 PRINT : PRINT "YOU JUST RAN INTO A
BRICK WALL!": PRINT
11710 QQ = INT ( RND (1) * 3 + 1) + QQ:
IF QQ < 7 THEN GOSUB 5360
11720 IF QQ > 10 THEN 11750
11730 A(LC) = B(LC): IF XX = 13 THEN LC
= LC - 20: GOTO 11740
```

```

11735 LC = LC + 20
11740 PRINT "THE MAGIC CHARIOT IS HURLED
        BACKWARDS!": GOTO 4250
11750 PRINT "THE MAGIC CHARIOT IS REDUCE
        D TO A PILE OF JUNK.": PRINT
11760 GOSUB 5000
11770 PRINT "YOU AND YOUR ENTIRE PARTY A
        RE DEAD.": PRINT : GOTO 1050
11780 PRINT "THE MAGIC CHARIOT HAS FLOWN
        INTO A THICK MYSTERIOUS FOG!"
11785 PRINT : PRINT
11790 FOR X=1 TO 160:PRINT"#";:NEXT:A(LC
        )=15:LC =INT(RND(1) * 50 + 1) + 30
11795 PRINT "THE FOG CLEARS.":ZZ = A(LC)
        :A(LC) = 12: GOTO 700
11800 PRINT : PRINT "A GIGANTIC BOULDER
        BLOCKS YOUR PATH!": PRINT
11810 R = INT ( RND (1) * 4 + 1)
11820 A(LC) = 27:ZZ = 27
11830 INPUT "WHICH WAY DO YOU TRY TO GO"
        ;M$:M$ = LEFT$ (M$,1)
11840 M = M + 1
11850 IF M$ = "D" AND R = 1 THEN 1000
11860 IF M$ = "U" AND R = 2 THEN 1190
11870 IF M$ = "R" AND R = 3 THEN 1200
11880 IF M$ = "L" AND R = 4 THEN 1220
11890 PRINT "THE BOULDER IS IN YOUR WAY!
        ": PRINT
11895 FOR X=1TO9:C(X)=C(X)-C(X)/20:NEXT:
        GOTO 11830

```

Table 9.1 Routines and Subroutines Used in THE GOLDEN FLUTE Program.

Routines

10-30	initialize
40-100	main preset
110-220	preset known map
230-590	preset hidden items
600-695	display introduction
700-890	display contents of the Magic Chariot
900-990	main play routine

1000-1010	move Down
1020-1050	out of bounds
1060-1140	end game/new game
1150-1175	win game
1190	move Up
1200-1210	move Right
1220-1230	move Left
4000-4020	display possible move key
4030-4220	fire Magic Bazooka
4250-4900	current location check

Subroutines

5000	timing delay loop
5010	check for blank location
5030-5115	instructions
5120-5290	visit Oracle
5300	foot of mountain display
5310-5350/5540-5570	infiltrate Terak's lair
5360-5530	character death
5580	check for out of bounds
5590-5595	"eerie singing in the distance" display
5600	find Magic Sword
5610-5620	find gold
5640-5650	find Magic Bazooka
5660-5740	Klufffoot/Friek attempts to re-steal Flute
5750-5980	Enchanted Forest/dead character revived
6000	find Magic Zither
6010-6040	enchanted land mine
6050-6080	wicked witch steals gold
7000-7250	display known map
8000-8010	clear maps
10000-10020	in the Woodlands message
10030-10070	Terak moves his lair
10080-10120	Magic Chariot mired in the Hopeless Pits
10130-10910	rub Magic Orb
10200-10230	Song of the Sirens
10240-10460	face monster
10490-10650	attempt to move past monster
10660-10995	toss gold overboard
11000-11490	hand to hand combat
11500-11620	unsheath Magic Sword
11630-11680	play Magic Zither
11700-11770	run into wall
11780-11795	mysterious fog
11800-11895	boulder

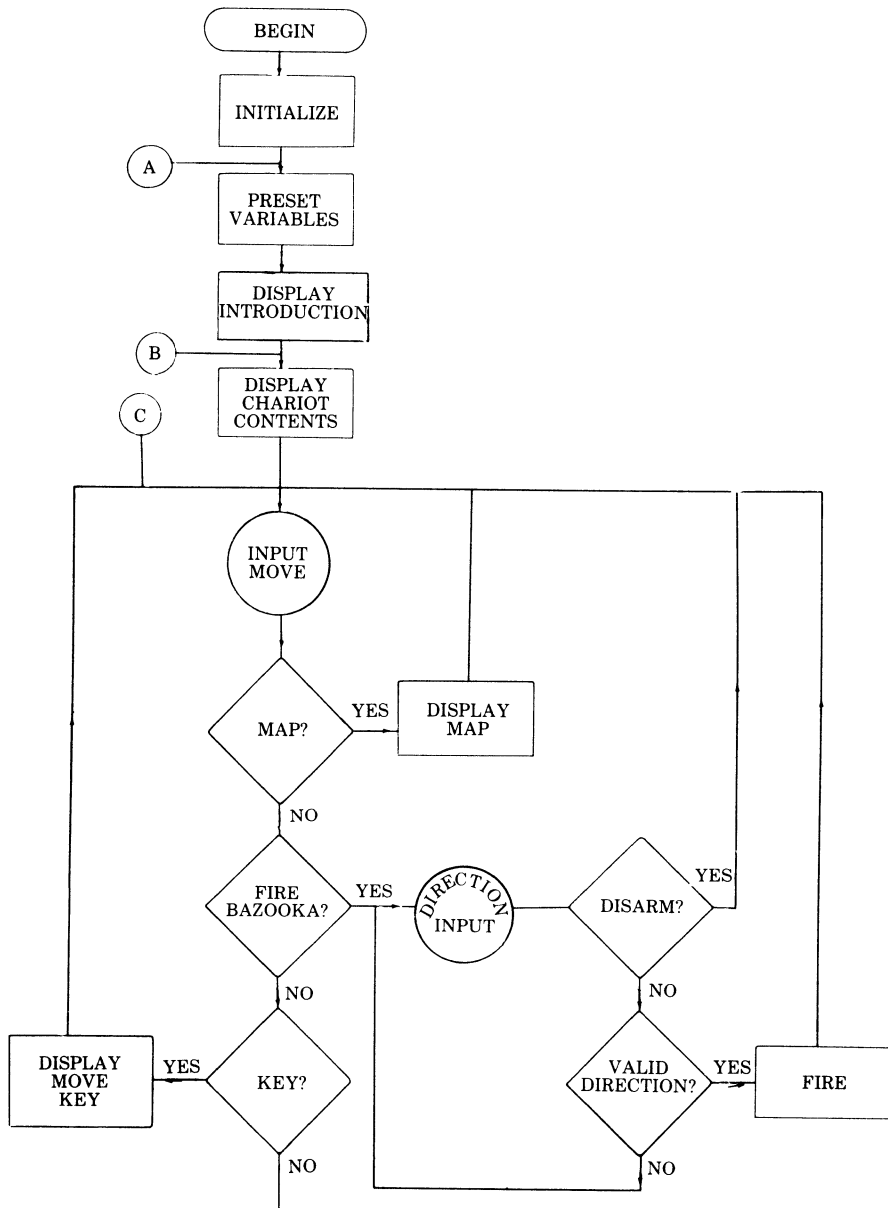


Figure 9.1A Flow-chart for THE GOLDEN FLUTE Program.

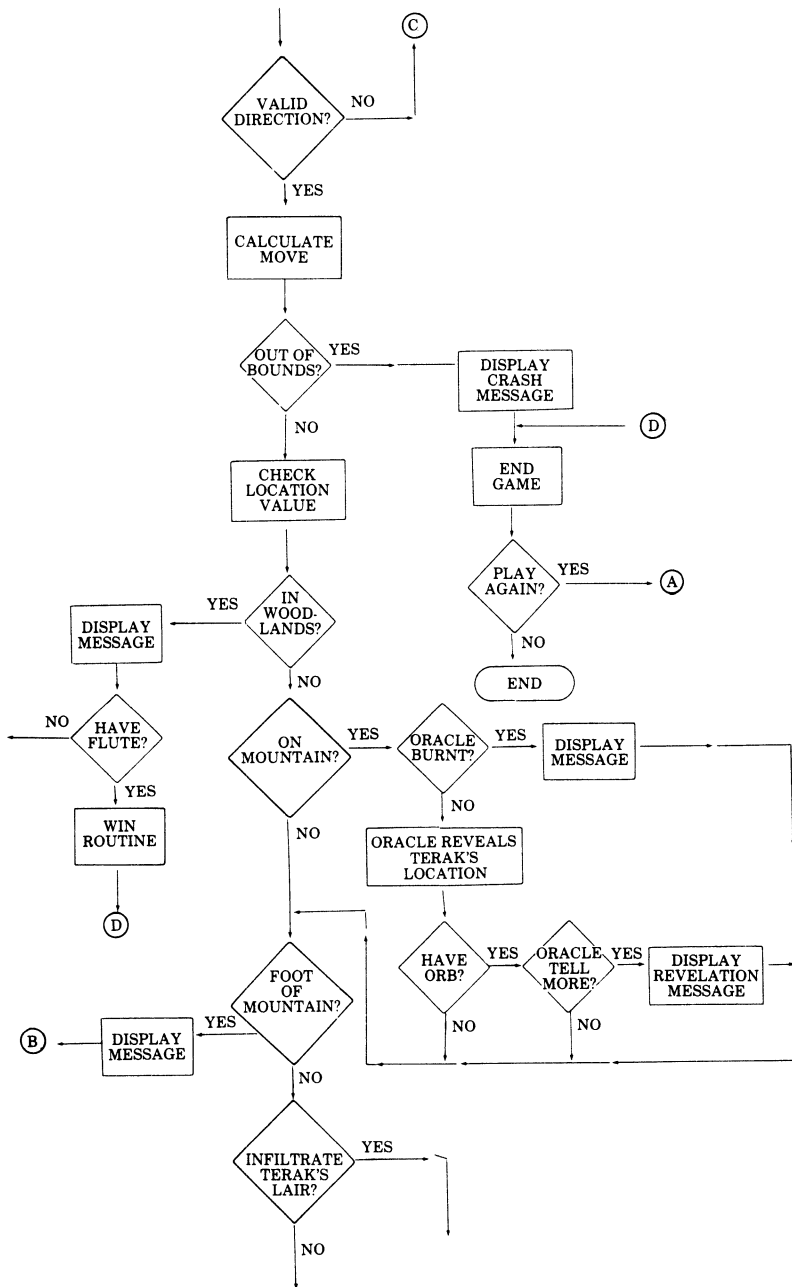


Figure 9.1B Flow-chart for THE GOLDEN FLUTE Program.

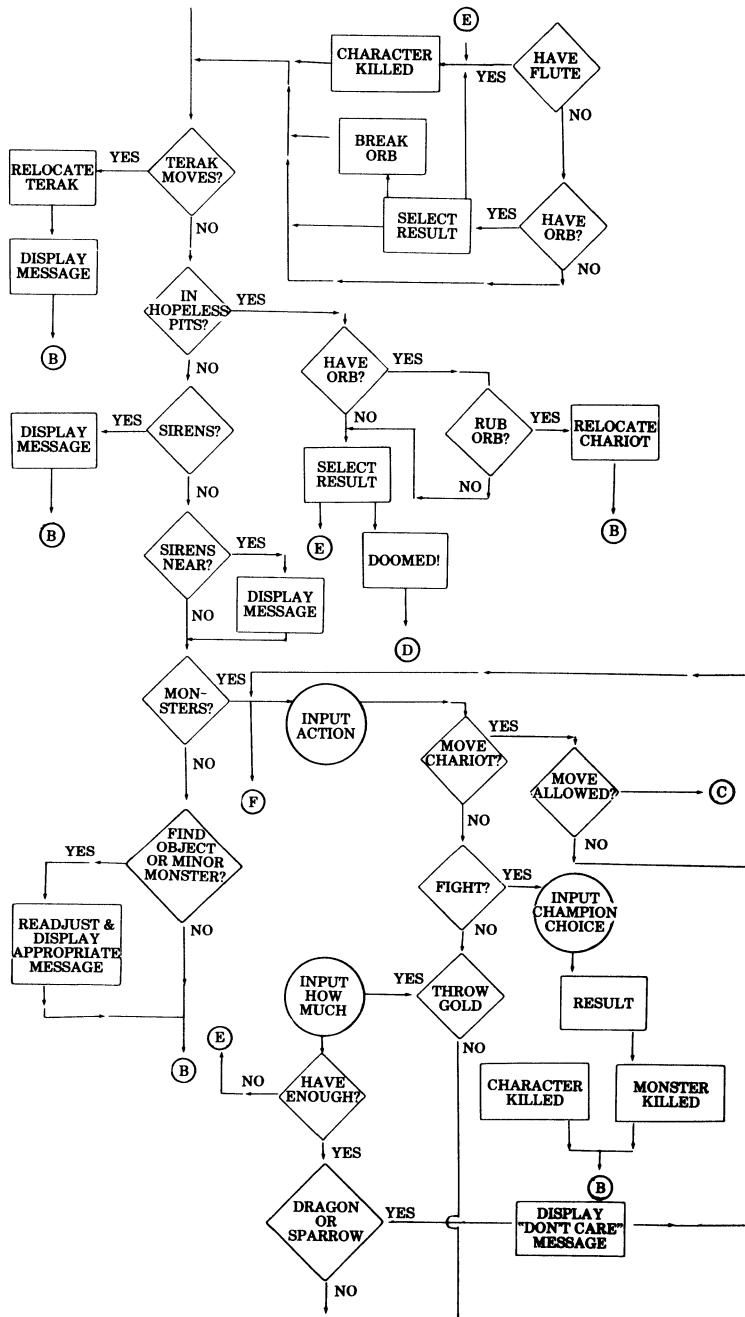


Figure 9.1C Flow-chart for THE GOLDEN FLUTE Program.

BR	Brombiran alive?
BS, BT, BU	bazooka firing limits
BZ	have Magic Bazooka?
CH	Champion chosen for hand to hand combat
D	character to be killed select
DG	Dagglette alive?
FL	have Golden Flute?
FO	monster currently being faced
G	bazooka firing limits
GL	golden coins
GR	Gromphlur alive?
GW	game counter
H	bazooka firing limits/gold to be tossed overboard/fight potential
I	character's fight power
JS	Jessan alive?
LC	current location
M	move counter
MO	have Magic Orb?
MZ	have Magic Zither?
P	Terak moves?action against monster
PM	Princess Melva alive?
PW	previous winning score
Q	misc.
QQ	number of collisions with wall
R	revive character select
RU	Rulf alive?
SJ	Sejjan alive?
SW	have Magic Sword?
TT	timing loop counter
UD, V, X	misc.
XX	current location value
XZ	display "*" location for explosion
Y, Z	misc.
ZX	crew count
ZZ	misc.

Arrays

A(100)	known map
B(100)	complete map
C(9)	character health ratings

String Variables

N\$	player's name
Q\$	various inputs

CHARACTERS

The mythical creatures who accompany the player on the Quest to recover the Golden Flute could remain anonymous and identified only by members, or by type (e.g., elf, satyr, fairy, etc.). But the details of the fantasy are always part of the fun of a good adventure game.

In the program, I have provided the player with eight companions, including three elves, a satyr, two fairies, the Queen of the Sprites, and a princess. These characters are identified by name in Table 9.3.

Table 9.3 Characters in THE GOLDEN FLUTE Game.

Aboard the Magic Chariot

N\$ "the Human" (player)
 "Brombiran, the Elf"
 "Dagglette, the Elf"
 "Gromphlur, the Elf"
 "Rulf, the satyr"
 "Jessan, the Fairy"
 "Sejjan, the Fairy"
 "Allegrecia, Queen of the Sprites"
 "Princess Melva"

Potential Helpers

Coramble, the Great Oracle
 dwarves

Villains

Terak (goal — has the Golden Flute)
 Klufffoot
 Friek
 gargoyles
 goblins
 dragons
 wicked witches
 Magic sparrows

Feel free to change any or all of the character names. They are listed at five points throughout the program (lines 730 through 800, 5440 through 5510, 5880 through 5950, 11010 through 11080,

and 11390 through 11460). For consistency, all five of these lists should be updated for each name change.

Perhaps you prefer to let the player assign the character names at the beginning of the program, using string variables.

THE PLAY

The playing area for THE GOLDEN FLUTE is similar to the one used in MARS. It is a 10-X-10 space map contained in arrays. Of course, this gives 100 possible places for the Magic Chariot to visit.

In MARS, if the player went past the boundaries of the map area, he looped around to the opposite end of the map. In THE GOLDEN FLUTE, a different approach is used. Leaving the legal boundaries of the defined playing area in this game results in an instant game loss. This is programmed in lines 1000, 1020 through 1050, and 1190 through 1220. If the player ever selects an out-of-bounds move, he forfeits the game, and the computer prints out the following message:

THE MAGIC CHARIOT HAS LEFT THE BOUNDARIES OF
THE MAGIC KINGDOM. WITHOUT MAGIC TO HOLD IT
UP, THE MAGIC CHARIOT CRASHES!

As in MARS, two maps of the playing area are set up in THE GOLDEN FLUTE program and two arrays—A(x) and B(x). This technique is extremely useful in a great many different adventure games.

Array A(x) contains information about explored and fixed areas of the Magic Kingdom. This is the map that is displayed in lines 7000 through 7250 of the program. Table 9.4 lists the various values that may be placed at each map location. Numbers that are not listed (for example, 16 or 18) are not used in the program as it is given here. You can use these unassigned values to set up your own additions to the game.

Table 9.4 also shows the characters used to indicate each item on the map. You'll notice that several items are marked only with a period (.). These are items that are invisible on the displayed map. It is up to the player to remember where these things are. Some items such as the Magic Sword and pieces of gold have no character symbol at all. These items are automatically loaded onto the Magic Chariot as soon as they are found, so the space is left empty. There is nothing to display. The space is treated like any blank location.

Table 9.4 Map Values Used in THE GOLDEN FLUTE Program.

value	meaning	displayed as
0	blank	
1	the Woodlands	W
2	Coramble, the Great Oracle of Purlicon Mountain	M
3	The Hopeless Pits	P
4	Terak's lair	T
5	Sirens	S
6	dragon	D
7	Magic Sword	.
8	gold coins	.
9	gargoyle	G
10	goblin	g
11	Magic Orb	.
12	The Magic Chriot	C
13, 14	wall	X
15	mysterious fog	F
16	undefined	.
17	Magic Bazooka	.
18, 19	undefined	.
20	smoldering ruin	*
21	Magic sparrow	s
22	ancient dwarf	d
23	Klufffoot	.
24	Friek	.
25	Enchanted Forest	.
26	magic Zither	.
27	boulder	R
28	enchanted land mine	!
29	wicked witch	w

Array B(x), on the other hand, is a complete map of the playing area. This is the map used by the computer whenever the player makes a move. The IF... THEN... statements in lines 4270 through 4900 are used to branch the program to the appropriate subroutine.

Throughout the game, a number of creatures may appear in the path of the Magic Chariot. These include ancient dwarfs, dragons, gargoyles, goblins, and magic sparrows. Each time one of these creatures is encountered, the computer displays a standardized list of possible actions for the player to choose from. Lines 10320 through 10380 display this list. If an invalid selection is made, the computer simply loops back around and asks for a new response.

The possible actions when facing one of these creatures include:

- 1—MOVE CHARIOT
- 2—THROW SOME GOLD OVERBOARD
- 3—HAND TO HAND COMBAT
- 4—RUB MAGIC ORB
- 5—UNSHEATHE MAGIC SWORD
- 6—PLAY MAGIC ZITHER

The first three choices (move chariot, throw gold, or combat) are always offered. The other three are only printed on the menu and accepted by the program if the appropriate object (Magic Orb, Magic Sword, or Magic Zither) is aboard the Magic Chariot.

Any of these actions will be useful in some circumstances. At other times they may work against the player's best interests. None of these actions will be a good choice in all cases.

Part of the appeal of adventure games is figuring out how to outmaneuver the creatures so the book won't help you here. However, a good adventure game is still fun to play, even after you've solved all the puzzles. Plenty of RND (random) statements in the program keep a game from going stale. But the player should use logic to deal with all obstacles, not just blind luck.

In any case, consider here a few details of each of the actions you could choose from when facing a creature in THE GOLDEN FLUTE.

If the player elects to move the Magic Chariot, he is queried for his desired direction. Sometimes a creature will allow the chariot to go one way but will block moves in all other directions. At other times, it will block all four directions. A few creatures don't care where the Chariot goes.

Each attempted move increments the move counter, adding to the player's score, even if the move is blocked off. Remember, THE GOLDEN FLUTE is scored like golf, with the object being to complete the game with as few moves as possible.

The second possible action is to throw pieces of gold overboard. Some creatures have no interest at all in gold. (A little logic should suggest which ones don't care.) Each of the other creatures determines how much gold it wants (a RND statement is used). If the player throws less than this amount of gold overboard, it will not help him. If the player equals or exceeds the creature's price, it is to the player's advantage. However, always bear in mind that it can be extremely dangerous to throw away more gold than you have.

The third possible action is hand-to-hand combat. When a player selects this alternative, the player is asked to pick a champion from among the characters aboard the Magic Chariot. He may not use a dead character.

Each character has a different strength rating. For example, the satyr is much stronger than either of the fairies. However, the monster (which has a randomly selected strength rating) is weakened by each battle. Jessan, the fairy, could conceivably slay a dragon that has just killed Rulf, the satyr.

The human (the player's character) is given the highest strength rating. But he should avoid going into hand-to-hand combat himself, unless everyone else in the Questing Party is already dead. If the player's character is killed, the game ends immediately.

Unsheathing the Magic Sword is generally pretty effective in dealing with creatures blocking the path of the Magic Chariot. Of course, the player must have already found the Magic Sword to use it.

Some gargoyles may have their own Magic Swords. A few dragons are immune to the Sword's magic. When the player unsheathes his Magic Sword before one of these creatures, he runs the risk of losing a Quester, and/or the Sword itself.

The Magic Orb can be used to get out of tough spots, but it can place the player into an even tougher spot.

Playing the Magic Zither usually doesn't accomplish much; but in certain circumstances, it proves worthwhile.

Occasionally, the Magic Chariot's path will be blocked by a giant boulder. Only one of the four directions may be used to leave the location. The player cannot always backtrack. If the player runs into a boulder by moving up, it may be in his way when he tries to move down. It is always possible to move in one direction. Unsuccessful move attempts are added to the move counter.

The player randomly selects the direction for passing a boulder each time the Chariot lands in a particular location. It can change if the player encounters the same boulder more than once in the same game. The boulders are as magical as everything else in this game.

If the player is on the edge of the playing area, a boulder can force him to move out of bounds, and he will have to forfeit the game. This could be very frustrating; but, fortunately, odds are that it will not happen often. Complicated and tricky programming would prevent this, but it may not be worth bothering with.

Avoiding the perimeters of the Magic Kingdom is part of the game strategy.

To land on the location occupied by the Sirens is instantly fatal. Whenever the Magic Chariot lands on a space adjacent to the Sirens (including diagonals), a warning message is displayed.

Hidden somewhere within the Magic Kingdom is a dandy weapon called the Magic Bazooka. The relevant programming is in lines 4030 through 4220 and 5640 through 5650. Once the player has found the Magic Bazooka, he can fire in any direction he chooses, instead of making a regular move. The Magic Bazooka cannot be used at close range, so it cannot be used when facing a monster.

When fired, the Magic Bazooka destroys everything in its path, leaving only smoldering ruins. It can destroy Coramble, the oracle; Terak; the Golden Flute, if he has it; the Woodlands; and any pieces of gold or hidden weapons. The Magic Bazooka must be used with considerable discretion.

PLAYING THE GAME

It's a good idea to start the game with a visit to Purlicon Mountain. Coramble, the Great Oracle of the Mountain, will reveal Terak's location (and possibly other useful information).

Occasionally Terak will fear the approach of the Magic Chariot and rehide his lair. When this happens, the player may have to return to the Oracle to find out the gremlin's new location.

If the player has the Magic Orb, Coramble may also reveal the location of some monsters. Then again, the ornery old cuss may not. This routine is programmed in lines 5120 through 5290.

SOME SECRETS OF THE GAME

You may want to enter the program and play the game a few times before reading the rest of this chapter. Solving the various problems you encounter is part of the fun of many adventure games. What should you do when faced by a gargoyle? What if a Magic Sparrow is in your path? The discussion in the next few pages reveals some solutions.

As Table 9.1 shows, the basic game is played in lines 700 through 1230. This section accepts the player's move and checks it for validity. There are six legal entries. They are K, U, D, R, L, and B. Only the first letter of the player's response is used by the program.

Entering K will print out a key to the legal moves.

Generally, the player responds with one of the four directional moves. If you enter U, the Magic Chariot moves up one space in the map. Similarly, an entry of D moves the Chariot down one space. R and L are used to move right and left. Remember that the player cannot move outside the boundaries of the map in any direction. No loop around is used in this game program.

If the player has located the Magic Bazooka, he has an additional possible choice. By entering B, he can fire the Magic Bazooka. See program lines 4030 through 4220. When B is entered, the computer will prompt the player for the desired direction. (He may change his mind and disarm the Magic Bazooka). The Bazooka then reduces everything in the selected direction to a smoldering ruin.

Once the player enters a directional move, the program checks the hidden complete map (array B(x)) for monsters, obstacles, weapons, or gold coins. These checks are performed in lines 4250 through 4900.

After each turn, the computer will ask the player if he wants a display map of the Magic Kingdom (lines 7000 through 7250). The program could easily be adapted to draw a map after each turn, but this slows the game down and becomes monotonous. It takes several seconds for the map to be displayed, and it can be irritating to wait when you don't particularly need it.

The characters of Table 9.4 are used in the display map. Unexplored locations are displayed as blanks.

THE MONSTERS

Since the creatures that confront the Magic Chariot provide much of the excitement of the game, it is worthwhile to note how they are dealt with in the program. The basic procedure is essentially the same for all five types of creatures—dragons, dwarfs, gargoyles, goblins, and Magic Sparrows. Now examine what happens when the Magic Chariot encounters a goblin.

Goblins are stored in the map arrays with the value 10. When the current location (B(LC)) equals 10 the program jumps from line 4470 to line 10260.

The first few steps at 10260 print out the type of monster (A GOBLIN), and the foe variable (FO) is set to a value of 3. This variable is used so that all the monsters can use the same choice of action routine (lines 10300 through 10460), yet separate results may be easily obtained for each type of creature.

The possible actions offered were discussed earlier. Now we will examine what happens with each option when the Magic Chariot is facing a goblin.

1. **MOVE CHARIOT.** In line 10300 a random number from 1 to 4 is generated. This number is assigned to the variable AM, and indicates the only move direction allowed. If AM equals 1, the Magic Chariot will only move up, and so forth.

When option 1 has been selected, the player is queried for his choice of direction (line 10500). The move counter is incremented, and the player's choice is compared to the value of AM. If the player has made the correct entry, the Chariot moves in the ordinary manner; and the monster will be displayed on future display maps. If, on the other hand, an unallowed move is made, the goblin will block the path of the Magic Chariot (lines 10610 through 10650), and the program jumps back to line 10320 to ask for another choice of action. Remember, the object of the game—return the Flute to the Woodlands in as few moves as possible. A move blocked by a monster is still counted. It isn't always the best strategy to simply try all four directions until the Chariot can get by the monster.

2. **THROW SOME GOLD OVERBOARD.** When this second option is selected, line 10660 asks the player how much gold he wants to throw. If he tries to throw more gold than he has aboard the Magic Chariot (checked in line 10680), a subroutine from lines 5360 through 5530 is called. The computer selects a random number from 1 to 8 in this subroutine. Each number represents one of the supporting characters aboard the Chariot. The selected character is killed (if he is already dead, nothing happens). The moral here is that you must always keep track of how much gold you are carrying. When facing a monster, you have no way to count your gold pieces.

If the player throws an acceptable number of gold coins overboard, his supply is appropriately decreased, and the goblin greedily eats the gold (line 10940). A random number from 1 to 200 is selected. If this number is less than the number of coins thrown, the goblin dies of terminal indigestion (line 10960) and is erased from both map arrays. If the goblin is not killed, the program returns to line 10320, and the player must select a new action.

3. **HAND-TO-HAND COMBAT.** This option was thoroughly discussed earlier. The goblin has a randomly determined strength rating. Either the goblin, or your chosen champion, will be slain.

The struggle will weaken the victor somewhat so it is a good idea to vary your champions. Even if your champion is slain, it may be well worthwhile to send out a second champion because the goblin has been weakened.

The outcome of the battle is determined in line 11340, which compares the strength rating of the opponents. The stronger fighter wins.

4. RUB MAGIC ORB. This option is offered in the menu and accepted by the program only if the Magic Orb is aboard the Magic Chariot (MO = 1).

Rubbing the Magic Orb randomly relocates the Chariot somewhere within the Magic Kingdom. Of course, since you could land in an even worse situation (for example, in the midst of the deadly Sirens), this option should be employed only in emergencies.

5. UNSHEATHE MAGIC SWORD. The player must possess the Magic Sword (SW = 1) for this action to be offered or accepted. Goblins are instantly slain by the Magic Sword. Other monsters may or may not be killed.

6. PLAY MAGIC ZITHER. Once again, the Magic Zither must be aboard the Chariot (MZ = 1) for this option to be used. Goblins tend not to be very fond of music. If this option is selected when facing a goblin (FO = 3—see line 11640), the subroutine for killing off a character (GOSUB 5360) is called; then a new action choice is requested. Obviously, the Magic Zither is not a very good way to deal with a goblin.

Actually, it is effective for only one type of monster. When the Zither is played for other creatures, the computer simply informs you that “No one is particularly impressed with your talent.”

The other monsters are programmed similarly to the goblin. Of course, specific results for each possible action may be different. Each creature must be treated individually. (Incidentally Magic Sparrows are surprisingly strong.)

EXPANDING THE GAME

You can expand this game program yourself in a number of ways. One of the most obvious expansions is to add more types of monsters. Use the undefined variable values in the map arrays to place these new creatures you devise. Refer back to Table 9.4 to see which values are already used in the present program.

Real-time actions could be called for when facing a monster. A timing loop and the GET command could be used as discussed in Chapter 7.

For the GOLDEN FLUTE game you could arrange the real-time programming so that if the player does not make a valid choice of action before the time loop runs out, one of the characters will be killed. The timing loop then starts over. If the player's character is the last one left alive aboard the Magic Chariot when the timing loop runs out, the program could jump to the YOU ARE DOOMED! and game message (line 10100).

If you need to conserve memory space, lines 710 through 890 could be eliminated. This section of the program simply prints out the current contents (personnel, weapons, and treasures) of the Magic Chariot after each turn. Since this routine does not directly affect the program in any way, it could be omitted without problems. However, the player must then always keep track of what he's carrying. This frustrates many people. If this section is eliminated, replace line 710 with a REMark statement. If no line exists at 710, the program will bomb.

If you want to speed up the map routine (lines 7000 through 7250), you could use PRINT CHR\$(V) statements, rather than the IF V = x THEN PRINT "XXX" format used in the current program. This would also tend to conserve memory space. You would have to change the values stored in the arrays for the appropriate character codes. The CHR\$(V) approach was not used here, because different codes would be required for different computers. Check your manual.

Graphic representations of the various monsters and obstacles could also be included in the program. Refer back to Chapter 7 for some hints in this area.

Another possible addition to the game would be to have the player bargain with Terak to get the Golden Flute back. If the player does not have enough coins, Terak keeps the Flute, and moves his lair.

The number of gold coins and/or live characters aboard the Magic Chariot when it returns the Golden Flute to the Woodlands affects the player's final score. This encourages more exploration of the Magic Kingdom (to find more gold) and more conservative strategy (to preserve characters).

Many other additions and plot twists are possible. As with all adventure type games, set your imagination free.

Chapter 10

The Great Escape

THE GREAT ESCAPE is a maze game. The player's character is lost in a 100-room building. The object, of course, is to find the way out. This program was written in stages that started with a simple maze game, then added a villain and gold coins. New objects and characters were then added on as they came to mind. Each item was completely programmed before work started on a new item. This prevents the possibility of getting lost and accidentally leaving out essential steps from the program.

Each room within the maze may have up to four doors (labeled North, South, East, and West). Some rooms may not have any doors at all. When the player gets stuck, he may use a secret passageway by entering X as his move. This will randomly relocate him within the maze. To discourage overuse of this feature, each secret passageway counts as 10 moves.

GOLD COINS AND SCORING

To add interest to the game, gold coins are scattered throughout most of the rooms. Any room may start out with anything from 0 to 10 coins. Weighting is used so that 0 will be the most common single value, yet more than half of the rooms should contain some coins. The number of coins in each room is stored in an array (GC(x)):

```
220 FOR X=1 TO 100:Y= INT (RND (1)*17+1)
230 IF Y > 10 THEN Y=0
240 GC(X)=Y: NEXT
```


The player's final score for the game is determined by how many gold coins he is carrying (as many as possible) and how many moves he used to escape from the building (as few as possible). There are one to three exits in the north-most wall (see line 130).

The scoring formula is calculated in line 5160. It is:

$$\text{INT} ((\text{TR}/\text{M}) * 100) - \text{M}$$

where TR represents the number of coins, and M is the number of moves.

For example, let's say the player escaped with 100 coins after 50 moves. The final score would equal $\text{INT} ((100/50)*100) - 50 = \text{INT} (2*100) - 50 = 200 - 50 = 150$.

Negative scores are also possible. For instance, if the player took 120 moves to get out of the building with 10 coins, his score would work out to $\text{INT} ((10/120)*100) - 120 = \text{INT} (0.083333 * 100) - 120 = 8 - 120 = 112$.

Generally, if the number of coins is less than the number of moves, the score will be negative.

In a series of games, the best previous positive score will be displayed at the end of each game. If the player's character is killed, the final score will not count, even if it is better than the previous high score.

OBSTACLES

As described so far, the game is functional, albeit a bit dull. A player is likely to soon lose interest. The solution, of course, is to add obstacles. If you glance over the complete program in Listing 10.1, you can see that most of the programming is devoted to the various obstacles and helpers. Just testing for the various additional characters and obstacles takes over 45 lines (from 650 to 1120).

Listing 10.1 Complete THE GREAT ESCAPE Program.

```
0 PRINT CHR$(147)
20 PRINT CHR$(147)
30 REM * THE GREAT ESCAPE * DELTON T. H
   ORN * V1.0
40 DIM T(10): DIM N(100): DIM S(100): DI
   M E(100): DIM W(100)
```

```
50 DIM BT(12): DIM CS(54): DIM CR(52): D
IM GC(100):G4 = 1
69 REM * SHUFFLE CARDS :
70 FOR X = 1 TO 52:CS(X) = 0:CR(X) = 0:
NEXT
80 FOR X = 1 TO 4: FOR Y = 1 TO 13
90 PRINT "*";Z = INT ( RND (1) * 52 +
1): IF CS(Z) > 0 THEN 90
100 CS(Z) = X:CR(Z) = Y + 1: NEXT : PRIN
T : NEXT
110 PRINT CHR$(147) : PRINT : PRINT : PR
INT "THE GREAT ESCAPE"
112 PRINT " ", "BY DELTON T. HORN"
119 REM * SET BUILDING *
120 FOR X = 1 TO 100:N(X) = 0:S(X) = 0:E
(X) = 0:W(X) = 0: NEXT
130 FOR X = 1 TO 3:Y = INT ( RND (1) *
10 + 1):N(Y) = 1: NEXT
140 FOR X = 1 TO 65:Y = INT ( RND (1) *
90 + 1):Z = Y + 10:N(Z) = 1
145 S(Y)=1:NEXT
150 PRINT : PRINT "YOU ARE LOST IN A GIA
NT MAZE OF          100 ROOMS!"
160 FOR X = 1 TO 73:Y = INT ( RND (1) *
99 + 1):Z = Y + 1:E(Z) = 1:W(Y) = 1
165 NEXT
170 PRINT "THE EXIT IS TO THE NORTH."
180 FOR X = 1 TO 10:YY = INT ( RND (1)
* 100 + 1):BT(X) = YY
190 Y = X * 10:Z = Y - 9:E(Z) = 0:W(Y) =
0: NEXT
200 PRINT : PRINT "FIND YOUR WAY OUT ---
";
210 FOR X = 1 TO 10:T(X) = INT ( RND (1)
) * 100 + 1): NEXT
220 FOR X = 1 TO 100:Y = INT ( RND (1)
* 17 + 1)
230 IF Y > 10 THEN Y = 0
240 GC(X) = Y: NEXT
250 PRINT "    IF YOU CAN!!!": PRINT
260 T5 = INT ( RND (1) * 100 + 1):K1 =
INT ( RND (1) * 60 + 1) + 30
```

```
265 IF T5 = K1 THEN 260
270 HA = 0:W1 = INT ( RND (1) * 100 + 1
):W2 = INT ( RND (1) * 100 + 1)
275 IF W1 = W2 THEN 270
280 G4 = G4 + 1:MF = 100:SC = INT ( RND
(1) * 100 + 1)
285 RB = INT ( RND (1) * 100 + 1): IF S
C = RB THEN 280
290 DF = 1:MC = INT ( RND (1) * 40 + 1)
+ 10
295 MG = INT ( RND (1) * 50 + 1) + 10:G
F = INT ( RND (1) * 50 + 1) + 40
300 IF (MC = MG) OR (MC = GF) OR (RB = M
C) OR (RB = MG) THEN 290
310 BX = 0:BB = INT ( RND (1) * 100 + 1
):G1 = INT ( RND (1) * 100 + 1)
315 BN = INT ( RND (1) * 100 + 1): IF G
1 = BN THEN 310
320 PRINT "IF YOU GET STUCK YOU CAN USE
A SECRET PASSAGEWAY BY ENTERING ";
330 CC = 1:H = 0:BT = 150:BR = INT ( RN
D (1) * 70 + 1) + 30
340 P = INT ( RND (1) * 50 + 1) + 40:M
= 1:V2 = 0
345 GH = INT ( RND (1) * 20 + 1) + 10
350 LM = INT ( RND (1) * 100 + 1): IF L
M = GH THEN 350
360 PRINT "'X' AS YOUR MOVE."
370 LQ = INT (RND (1) * 40 + 1) + 60:QW =
ABS (LQ - P): IF QW < 15 THEN 370
380 ML = INT ( RND (1) * 90 + 1) + 10:L
2 = INT ( RND (1) * 80 + 1) + 6
385 IF L2 = ML THEN 380
390 V = INT ( RND (1) * 100 + 1): IF V
= P THEN 390
400 PRINT "A SECRET PASSAGEWAY WILL COUN
T AS 10 MOVES."
410 N = 0:GN = INT ( RND (1) * 90 + 1)
+ 10.6:LH = 0
415 LQ = INT ( RND (1) * 45 + 1) + 55:S
W = INT ( RND (1) * 100 + 1)
```

```
420 PD = INT ( RND (1) * 100 + 1):K =  
INT ( RND (1) * 40 + 1) + 60  
425 S = INT ( RND (1) * 90 + 1) + 10  
430 PRINT : INPUT "WOULD YOU LIKE TO TRY  
A HARD GAME";Q$  
440 Q$ = LEFT$ (Q$,1)  
450 IF Q$ = "Y" THEN GOSUB 10000  
499 REM * THE MAIN PLAY *  
500 IF P > 100 THEN P = INT ( RND (1) *  
10 + 1) + 90  
510 PRINT CHR$(147) : PRINT : PRINT : PR  
INT "MOVE #";M;" ";  
520 IF LH = 0 THEN PRINT "YOU ARE NOW I  
N ROOM #";P: GOTO 530  
525 PRINT  
530 O = 0: FOR U = 1 TO 10: IF BT(U) = P  
THEN O = 1  
540 IF O = 1 THEN GOSUB 10030  
550 IF V > 100 THEN V = INT ( RND (1) *  
20 + 1) + 80  
555 PRINT  
560 IF TR = 0 THEN 580  
570 PRINT "YOU ARE CARRYING ";TR;" GOLD  
COIN";: IF TR > 1 THEN PRINT "S"  
571 IF TR > 1 THEN 580  
575 PRINT  
580 IF V < 1 THEN 650  
590 GOSUB 10180: IF V = P THEN 5000  
600 PRINT "HE MOVES TO FIND YOU!":W = I  
NT ( RND (1) * 10 + 1)  
610 IF V > P THEN V = V - W: GOTO 620  
615 V = V + W  
620 GOSUB 10180  
630 IF V = P THEN 5000  
649 REM * SPECIAL TESTS *  
650 PRINT : IF BX = 0 THEN 680  
660 PRINT "YOU ARE CARRYING A MYSTERIOUS  
LY TICKING BOX."  
670 IF (BX < M) OR (BX = M) THEN 5220  
680 IF SC = 0 THEN PRINT "YOU ARE CARRY  
ING A SCREWDRIVER."
```

```

690 IF SW = 0 THEN PRINT "YOU ARE CARRY
ING A SAW."
700 IF BN = P THEN GOSUB 10530
710 IF BN = 0 THEN GOSUB 10570
730 IF K1 = 0 THEN PRINT "YOU ARE CARRY
ING A KEY."
740 IF BB = P THEN GOSUB 10270
750 IF SC = P THEN PRINT "YOU JUST FOUN
D A SCREWDRIIVER.":SC = 0
760 IF K1 = P THEN PRINT "YOU JUST FOUN
D A KEY ":K1 = 0
779 IF GC(P) = 0 THEN 810
780 PRINT "YOU JUST FOUND ";GC(P);" GOLD
COIN";
790 IF GC(P) > 1 THEN PRINT "S";
800 PRINT "!":TR = TR + GC(P):GC(P) = 0
810 IF SW = P THEN GOSUB 10310
820 IF GN < 1 THEN GOSUB 10390
830 IF INT (GN) = P THEN GOSUB 10350
840 IF K = P THEN GOSUB 10420
850 IF TR=P THEN PRINT "YOUR GOLD COINS
ARE MAGICALLY DOUBLED!":TR = 2 * TR
860 IF LM = P THEN 5250
870 IF L2 = P THEN 5450
880 X = INT ( RND (1) * 45 + 1)
885 IF ((X > 40) AND (LH = 0)) OR ((X >
30) AND (LH = 1)) THEN GOSUB 10500
890 IF (N > 0) AND((N = M) OR (N < M)) T
HEN PRINT "IT'S TOO LATE!": GOTO 5130
900 IF N>0 THEN PRINT "YOU MUST FIND THE
FIRST AID KIT WITHIN ";N-M;" MOVES."
910 IF P = PD THEN 5620
920 IF P = - PD THEN PRINT "THERE IS A
DEAD PUPPY DOG IN THIS ROOM."
930 IF BT = P THEN GOSUB 10620
940 IF DF = P THEN GOSUB 10650
950 IF MF = P THEN GOSUB 10670
960 DF = DF + 1:MF = MF - .5
970 IF P = GF THEN GOSUB 10700
980 IF P = S THEN GOSUB 5800
990 IF P = - RB THEN 995 ELSE 1000

```

```
995 PRINT "THERE IS A PILE OF USELESS ME  
TALLIC      JUNK ON THE FLOOR."  
1000 IF P = RB THEN 5900  
1010 IF P = MG THEN 6450  
1020 IF P = MC THEN 6680  
1030 IF P = GH THEN 7000  
1040 IF P = G1 THEN 7150  
1050 IF P = W1 THEN 7430  
1060 IF P = W2 THEN 7460  
1070 IF P = T5 THEN 7600  
1075 IF P = - T5 THEN PRINT "THERE IS  
AN EMPTY TREASURE CHEST HERE."  
1080 IF LQ = 0 THEN 1090  
1090 IF LQ = P THEN GOSUB 10920: GOTO 1  
100  
1095 GOSUB 10960  
1100 IF (BT > 0) AND (BT < 25) THEN PRI  
NT "YOUR FLASHLIGHT IS GETTING DIM."  
1110 VS = INT ( RND (1) * 7 + 1)  
1115 IF (VS>5) AND (V>0) THEN PRINT "TH  
E VILLAIN IS NOW LURKING IN ROOM #";V  
1120 GOSUB 10750  
1130 PRINT : PRINT "          *** AVAILABL  
E EXITS ***"  
1135 PRINT  
1140 IF BT < 1 THEN PRINT "????",: GOTO  
1190  
1150 IF N(P) = 1 THEN PRINT "NORTH  ";  
1160 IF S(P) = 1 THEN PRINT "SOUTH  ";  
1170 IF E(P) = 1 THEN PRINT "EAST   ";  
1180 IF W(P) = 1 THEN PRINT "WEST   ";  
1190 IF SW = 0 THEN PRINT "CUT NEW EXIT  
";  
1195 PRINT  
1200 M$ = "":PRINT : PRINT " ",: INPUT "  
YOUR MOVE";M$:M$ = LEFT$ (M$,1)  
1205 PRINT  
1210 M = M + 1  
1220 IF M$ = "X" THEN 4000  
1230 IF (M$ = "N") AND (N(P) = 1) THEN 4  
020
```

```
1240 IF (M$ = "S") AND (S(P) = 1) THEN 4
030
1250 IF (M$ = "E") AND (E(P) = 1) THEN 4
040
1260 IF (M$ = "W") AND (W(P) = 1) THEN 4
050
1270 IF (M$ = "C") AND (SW = 0) THEN 406
0
1280 HA = HA + 1: PRINT "YOU JUST RAN IN
TO A WALL, KLUTZ!"
1290 IF HA < 50 THEN 520
1300 PRINT "YOU HAVE WALKED INTO SO MANY
WALLS, YOU SUFFER BRAIN DAMAGE!"
1310 IF N > 0 THEN 5130
1320 N = M + 75: PRINT "YOU MUST FIND TH
E FIRST AID KIT WITHIN"
1325 PRINT " 75 MOVES OR YOU WILL LAPSE
INTO A          PERMANENT COMA."
1330 GOTO 520
4000 PRINT : PRINT "          SECRET PASS
AGEWAY": PRINT : PRINT
4010 BT = BT - 9: FOR X = 1 TO 222: NEXT
X
4015 P = INT ( RND (1) * 100 + 1):M = M
+ 9: GOTO 500
4020 P = P - 10: IF P < 1 THEN 4200
4025 GOTO 500
4030 P = P + 10: GOTO 500
4040 P = P - 1: IF P < 1 THEN 4200
4045 GOTO 500
4050 P = P + 1: GOTO 500
4060 INPUT "WHICH WALL DO YOU CUT A HOLE
IN";Q$:Q$ = LEFT$ (Q$,1)
4070 IF (Q$ = "N") OR (Q$ = "S") OR (Q$
= "E") OR (Q$ = "W") THEN 4090
4080 PRINT "THERE IS NO SUCH WALL.": GOT
O 1130
4090 X = INT ( RND (1) * 20 + 1): IF X
> 15 THEN 4150
4100 IF Q$ = "N" THEN N(P) = 1
4110 IF Q$ = "S" THEN S(P) = 1
4120 IF Q$ = "E" THEN E(P) = 1
```

```
4130 IF Q$ = "W" THEN W(P) = 1
4140 IF X > 9 THEN 4170
4145 GOTO 1130
4150 PRINT "THE SAW BREAKS BEFORE YOU MA
KE AN"
4155 PRINT "OPENING LARGE ENOUGH TO CRAW
L THROUGH."
4160 SW = - 1: GOTO 1130
4170 PRINT "THE SAW BREAKS JUST AS YOU F
INISH      CUTTING THE NEW EXIT."
4180 SW = - 1: GOTO 1130
4200 PRINT " ", "YOU MADE IT!!!"
4210 PRINT "IT TOOK YOU ";M;" MOVES,":PRI
NT "AND YOU GOT OUT WITH ";TR;" COINS!"
4220 GOTO 5160
4998 STOP
4999 REM * VILLAIN ENCOUNTER *
5000 IF GN < 1 THEN 5050
5010 IF H = 1 THEN 5080
5020 PRINT "HE CAPTURES YOU AND THROWS Y
OU THROUGH  A SECRET PASSAGEWAY!"
5030 IF TR > 0 THEN PRINT "HE ALSO STEA
LS ALL OF YOUR GOLD COINS!"
5035 IF TR > 0 THEN V2 = V2 + TR:TR = 0
5040 P = INT ( RND (1) * 10 + 1) + 90:
GOTO 650
5050 INPUT "DO YOU SHOOT AT HIM";Q$:Q$ =
LEFT$ (Q$,1)
5060 X = 0: IF Q$ = "Y" THEN 5070
5065 GOTO 5010
5070 GOSUB 10210:H = 1: IF X > 6 THEN 52
10
5075 PRINT "YOU MISSED!": GOTO 5010
5080 PRINT "HE PULLS OUT HIS OWN GUN AND
SHOOTS AT YOU!":FOR X=1 TO 222:NEXT
5090 PRINT " ", "*** BANG! ***":FOR X=1 TO
222:NEXT:PRINT:X=INT(RND(1)*10 + 1)
5100 IF X > 7 THEN 5120
5110 PRINT "WHEW!  HE MISSED!": GOTO 50
20
5120 PRINT "HE GOT YOU!": PRINT
5130 PRINT "YOU ARE DECEASED...": PRINT
```



```
5140 PRINT:PRINT"YOU SURVIVED ";M;" MOVE
S,":PRINT"AND HAD ";TR;" GOLD COINS."
5150 RT = 0: GOTO 5170
5160 RT = INT ((TR / M) * 100) - M: PRI
NT "YOUR SCORE THIS TIME WAS ";RT
5170 IF RX = 0 THEN RX = RT
5180 PRINT "PREVIOUS HIGH SCORE WAS ";RX
5190 IF RT > RX THEN RX = RT
5200 PRINT "PLAY GAME #";G4;:INPUT Q$:Q$
=LEFT$(Q$,1): IF Q$ = "Y" THEN 110
5205 KQ$="LOF$,8"+CHR$(13)+"RU"+CHR$(13)
:F$="MENU"
5206 FOR I=631 TO 640
5207 POKE I,ASC(MID$(KQ$,I-630,1))
5208 NEXT I
5209 POKE 198,10:END::REM 10 CHARS IN QU
EUE
5210 PRINT "GOT 'EM!!":V = 0: GOTO 650
5220 PRINT "THE TIME BOMB IN THE MYSTERI
OUSLY TICKING BOX EXPLODES!"
5230 FOR X = 1 TO 222: NEXT : PRINT " ",
"KA--";
5240 FOR X = 1 TO 234: NEXT : PRINT "BOO
M!!": PRINT : GOTO 5130
5249 REM * MAGIC LAMP 1 *
5250 GOSUB 10450
5260 IF Q$ = "Y" THEN 5280
5270 GOTO 870
5280 LM = LM + INT ( RND (1) * 5 + 1):M
L = INT ( RND (1) * 9 + 1)
5300 IF (ML = 1) AND (GN < 1) THEN 5280
5310 IF ML=1 THEN PRINT"YOU ARE TRANSPOR
TED TO THE ROOM WITH THE GUN."
5315 IF ML=1 THEN P = INT (GN)
5320 IF ML=2 THEN PRINT"YOU ARE TRANSPOR
TED TO THE ROOM WITH"
5325 IF ML = 2 THEN PRINT "THE FIRST AI
D KIT.":P = K
5330 IF (ML = 3) AND (SW < 1) THEN 5280
5340 IF ML=3 THEN PRINT"YOU ARE TRANSPOR
TED TO THE ROOM WITH THE SAW.":P=SW
```

```
5350 IF ML = 4 THEN PRINT "THE VILLAIN
IS PLACED IN ROOM #100.":V = 100
5360 IF ML=5 THEN P=INT(RND(1)*20+1):PRI
NT "YOU ARE TRANSPORTED TO ROOM #";P
5370 IF (ML = 6) AND (GH = 0) THEN 5280
5380 IF ML = 6 THEN PRINT "THE GHOST HA
S BEEN EXORCISED.":GH = 0
5390 IF ML = 7 THEN PRINT "YOUR MOVE CO
UNT HAS BEEN REDUCED."
5395 IF ML = 7 THEN M = INT (M - INT
( RND (1) * M / 4 + 1))
5400 IF (ML = 8) AND (LH = 0) THEN 5280
5410 IF ML = 8 THEN PRINT "A MAP APPEAR
S.":LH = 0
5420 IF ML = 9 THEN PRINT "A SET OF SPA
RE BATTERIES APPEARS.":BT = BT + 100
5430 PRINT : INPUT "PLEASE PRESS 'RETURN
' ";Q$
5440 GOTO 500
5450 GOSUB 10450
5460 IF Q$ = "Y" THEN 5480
5470 GOTO 880
5480 L2 = 0:ML = INT ( RND (1) * 12 + 1
)
5490 IF ML = 1 THEN PRINT "DAME FORTUNE
IS DEAD.":DF = 101
5500 IF ML = 2 THEN PRINT "MISS FORTUNE
IS DEAD.":MF = 0
5510 IF ML = 3 THEN PRINT "THE LEPRECHA
UN IS DEAD.":LQ = 0
5520 IF ML = 4 THEN PRINT "THE VILLAIN
IS DEAD.":V = 0
5530 IF ML = 5 THEN PRINT "THE ROBOT'S
BATTERIES ARE DEAD.":RB = 0
5540 IF ML = 6 THEN PRINT "THE MAD GAMB
LER IS DEAD.":MG = 0
5550 IF ML=7 THEN PRINT "THE GHOST HAS B
EEN RETURNED TO ITS GRAVE."
5555 IF ML=7 THEN GH = 0
5560 IF ML = 8 THEN PRINT "THE EVIL MER
CHANT IS DEAD.":MC = 0
```

```
5570 IF ML = 9 THEN PRINT "THE PUPPY DO  
G IS DEAD.":PD = - PD  
5580 IF ML = 10 THEN PRINT "THE GORILLA  
IS DEAD.":G1 = 0  
5590 IF ML = 11 THEN PRINT "THE GOOD FA  
IRY IS DEAD.":GF = 0  
5600 IF ML = 12 THEN 5130  
5610 GOTO 880  
5620 PRINT "THERE IS A PUPPY DOG IN THIS  
ROOM.":X = INT ( RND (1) * 10 + 1)  
5630 IF X > 7 THEN PRINT " ", "WOOF! WO  
OF!"  
5640 IF GN > .99 THEN 5710  
5650 INPUT "DO YOU SHOOT THE PUPPY DOG";  
Q$:Q$ = LEFT$ (Q$,1)  
5660 IF Q$ = "Y" THEN 5680  
5670 GOTO 5710  
5680 PRINT "OH, YOU ARE A MEAN PERSON.":  
GOSUB 10210  
5690 IF (X > 0) AND (X < 7) THEN PD = -  
PD: GOTO 910  
5700 PRINT "YOU MISSED.":Y = 10  
5710 X = INT ( RND (1) * Y + 1): IF X <  
4 THEN 5740  
5720 IF X < 7 THEN PRINT "THE PUPPY DOG  
WAGS ITS CUTE LI'L TAIL."  
5730 GOTO 920  
5740 PRINT "THE PUPPY DOG BITES YOU!": I  
F N > 0 THEN 5760  
5750 N = 100 + M: GOTO 5780  
5760 X = N - M: IF X < 4 THEN 5130  
5770 N = M + INT (X / 2)  
5780 PRINT "YOU MUST FIND THE FIRST AID  
KIT WITHIN ";N - M;" MOVES."  
5790 GOTO 920  
5800 PRINT "THERE IS A SNAKE IN THIS ROO  
M.": PRINT " ", "HSSSS...": PRINT  
5810 IF GN < 1 THEN 5860  
5820 PRINT "YOU ARE BITTEN!": IF N > 0 T  
HEN 5130  
5830 PRINT "THERE IS SOME ANTITOXEN IN TH  
E FIRST AID KIT. "
```

```
5835 PRINT"YOU MUST FIND IT WITHIN 50 MO
VES OR DIE.":N = M + 30
5850 X = INT ( RND (1) * 10 + 1)
5855 IF X>5 THEN PRINT"THE SNAKE SLITHER
S OFF.":S = INT ( RND (1) * 100 + 1)
5860 INPUT "DO YOU TRY TO SHOOT THE SNAK
E":Q$:Q$ = LEFT$ (Q$,1)
5870 IF Q$ = "Y" THEN GOSUB 10210: GOTO
5880
5875 GOTO 5820
5880 IF X > 5 THEN PRINT "YOU MISSED!":
GOTO 5820
5890 PRINT "GOT 'EM!":S = 0: GOTO 990
5900 PRINT "THERE IS AN EIGHT FOOT TALL
ROBOT IN THIS ROOM!"
5910 PRINT " ", "* BEEP *", " ", "* BEEP *"
: PRINT
5920 RG = INT ( RND (1) * 50 + 1)
5930 PRINT:PRINT " WHAT DO YOU DO?":
PRINT "X -- USE SECRET PASSAGEWAY"
5940 IF BN = 0 THEN PRINT "F -- FEED BA
NANAS TO ROBOT"
5950 IF GN < 1 THEN PRINT "S -- SHOOT R
OBOT"
5960 PRINT "P -- PUSH FAST ROBOT TO NEAR
EST EXIT"
5970 IF SC = 0 THEN PRINT "D -- DISMANT
LE ROBOT"
5980 IF TR > 0 THEN PRINT "G -- GIVE SO
ME GOLD COINS TO THE ROBOT"
5990 PRINT "C -- CRY": PRINT "E -- EAT R
OBOT"
6000 INPUT "YOUR SELECTION":Q$:Q$ = LEF
T$ (Q$,1)
6010 IF Q$ = "X" THEN 4000
6020 IF (Q$ = "F") AND (BN = 0) THEN 610
0
6030 IF (Q$ = "S") AND (GN < 1) THEN 613
0
6040 IF Q$ = "P" THEN 6170
6050 IF (SC = 0) AND (Q$ = "D") THEN 629
0
```

```
6060 IF (TR > 0) AND (Q$ = "G") THEN 6320
6070 IF Q$ = "C" THEN 6400
6080 IF Q$ = "E" THEN 6430
6090 PRINT "THAT IS NOT AN ACCEPTABLE ACTION.": GOTO 5930
6100 PRINT "THE ROBOT TAKES THE BANANAS IN ITS GRIPPER ";
6110 FOR X = 1 TO 232 : NEXT
6115 PRINT "AND SQUEEZES THEM INTO A DISGUSTING PULP."
6120 FOR X=1 TO 246:NEXT:PRINT:PRINT "ROBOTS DO NOT EAT BANANAS.": GOTO 5930
6130 GOSUB 10210: IF X = 0 THEN 5930
6140 PRINT "THE BULLET BOUNCES OFF THE ROBOT'S"
6145 PRINT "METALLIC HIDE WITHOUT EVEN DENTING IT."
6150 IF X < 9 THEN 5930
6160 PRINT "BUT IT HITS YOU ON THE RICHOTCHET!": GOTO 5130
6170 PRINT "THE ROBOT WILL NOT LET YOU PUSH BY.": PRINT "IT BREAKS YOUR ";
6180 X = INT ( RND (1) * 7 + 1): IF X = 1 THEN PRINT "ARM":Y = 100:XY = 5
6190 IF X = 2 THEN PRINT "LEG":Y = 75:X Y = 8
6200 IF X = 3 THEN PRINT "NOSE":Y = 80:XY = 6
6210 IF X = 4 THEN PRINT "THUMB":Y = 150:XY = 2
6220 IF X = 5 THEN PRINT "BACK":Y = 35:XY = 17
6230 IF X = 6 THEN PRINT "NECK":Y = 30:XY = 20
6240 IF X = 7 THEN PRINT "BIG TOE":Y = 150:XY = 3
6250 IF N > 0 THEN 6280
6260 N=Y+M:PRINT"YOU MUST FIND THE FIRST AID KIT WITHIN ";
6265 PRINT N-M;" MOVES."
6270 GOTO 5930
```

```
6280 N = N - XY: IF N < M THEN 5130
6290 PRINT "THE SCREWDRIVER COMES IN HAN
DY. YOU SOONREDUCE THE ROBOT TO A ";
6300 PRINT "HEAP OF USELESS JUNK.":RB
= - RB
6310 GOTO 1010
6320 PRINT "HOW MANY COINS DO YOU OFFER
THE ROBOT";
6325 INPUT X
6330 IF X > TR THEN 6370
6340 TR = TR - X: PRINT "THE ROBOT TAKES
";X;" GOLD COINS AND ";
6350 IF X > RG THEN PRINT "LEAVES":RB =
- RB: GOTO 1010
6360 PRINT "BLINKS A FEW LIGHTS AT YOU."
: GOTO 5930
6370 PRINT "YOU DO NOT HAVE ";X;" COINS!
"
6380 TR = 0: PRINT "THE ROBOT TAKES THE
COINS YOU DO HAVE."
6390 PRINT "THEN IT BREAKS YOUR ";: GOTO
6180
6400 PRINT "CRYING WILL DO YOU NO GOOD."
6410 PRINT "DID YOU THINK THE ROBOT WOULD
TAKE PITY ON YOU?"
6420 PRINT "HA! IT HAS A HEART OF TIN."
: GOTO 5930
6430 PRINT"YOU CAN'T EAT A ROBOT, YOU SLL
LY PERSON.":X=INT(RND(1)*12+1)+1
6440 PRINT "YOU ONLY SUCCEED IN BREAKING
";X;" TEETH.": GOTO 5930
6450 PRINT "YOU JUST MET UP WITH THE MAD
GAMBLER!": PRINT
6460 INPUT "PLEASE PRESS 'RETURN' ";Q$
: PRINT CHR$(147):IF TR>10 THEN 6490
6470 PRINT "BECAUSE YOU HAVE SO FEW COIN
S, HE HAS":P=INT(RND(1)*20+1) + 80
6475 PRINT "NO INTEREST IN YOU. HE THROW
S YOU THROUGH A TRAP DOOR."
6485 INPUT "PLEASE PRESS 'RETURN' ";Q
$: GOTO 510
6490 PRINT "MAD GAMBLER: HE-HE-HEE! GR
```

```
EETINGS,"
6495 PRINT "MY PIGEON! WE SHALL PLAY A L
      ITTLE HIGH"
6496 PRINT "CARD.  IF YOU DRAW THE HIGH
      CARD, I'LL"
6497 PRINT "LET YOU EXIT NORTH, AND I'LL
      DOUBLE"
6530 PRINT "YOUR GOLD COINS.  BUT IF I W
      IN, I GET"
6535 PRINT "HALF YOUR COINS, AND WE PLAY
      AGAIN."
6550 FOR X = 1 TO 234: NEXT : PRINT : PR
      INT "YOU DRAW -- ";
6560 GOSUB 10820
6570 X1 = CS(CC):X2 = CR(CC):CC = CC + 1
6580 GOSUB 10820
6590 Y1 = CS(CC):Y2 = CR(CC):CC = CC + 1
6600 IF X1 > Y1 THEN 6650
6610 IF (X1 = Y1) AND (X2 > Y2) THEN 665
      0
6620 PRINT"  YOU LOSE!!!":PRINT"MAD GAMB
      LER:  HA! HA! HA!":TR=INT(TR/2)
6625 IF TR < 8 THEN 6470
6630 PRINT "LET'S PLAY AGAIN, PIGEON.":
      INPUT "PRESS 'RETURN' TO DRAW  ";Q$
6635 GOTO 6550
6650 PRINT "  YOU WIN!!!": PRINT "MAD GA
      MBLER:  RATS!"
6660 P = P - 10:TR = TR * 2: INPUT "PRES
      S 'RETURN' TO EXIT TO THE NORTH  ";Q$
6670 GOTO 500
6680 PRINT "YOU ENCOUNTER AN EVIL MERCHA
      NT.": IF TR < 10 THEN 6470
6690 PRINT "EVIL MERCHANT:  WELL. MY FRI
      END -- YOU "
6695 PRINT "MAY EXIT TO THE SOUTH, OR YO
      U CAN  "
6697 PRINT "BARGAIN WITH ME FOR AN EXIT
      TO THE  "
6698 PRINT "NORTH. DO YOU WANT TO BARGAI
      N";
6720 INPUT Q$:Q$ = LEFT$(Q$,1)
```

```
6730 IF Q$ = "Y" THEN 6770
6740 PRINT
6745 PRINT "EVIL MERCHANT:  I'M SORRY WE
COULDN'T  DO BUSINESS TOGETHER."
6750 PRINT : INPUT "PRESS 'RETURN' TO EX
IT TO THE SOUTH";Q$
6760 P = P + 10: GOTO 500
6770 X = INT ( RND (1) * TR + 1):CO = T
R
6780 PRINT : PRINT "EVIL MERCHANT:  MAKE
ME AN OFFER"
6785 PRINT "MY FRIEND.  (ENTER 0 TO END
BARGAINING)"
6800 INPUT "YOUR OFFER";Y: IF Y = 0 THEN
6740
6805 IF Y > TR THEN 6840
6810 IF Y < X THEN 6880
6820 PRINT "EVIL MERCHANT:  ALL RIGHT.
I'LL ACCEPT THAT, MY FRIEND."
6830 P=P-10:INPUT"PRESS 'RETURN' TO EXIT
TO THE NORTH  ";Q$:TR=TR-Y:GOTO 500
6840 PRINT "EVIL MERCHANT:  BAH! YOU DO
NOT HAVE": PRINT Y;" GOLD COINS!"
6850 PRINT "  I DO NOT ENJOY BEING CHEAT
ED!": PRINT
6860 PRINT "HE TAKES WHAT COINS YOU DO H
AVE AND"
6865 PRINT "THROWS YOU THROUGH A TRAP DO
OR!"
6870 TR=0:P=INT(RND(1)*20+1)+80:INPUT"
PRESS 'RETURN'  ";Q$: GOTO 500
6880 PRINT "EVIL MERCHANT:  ";:XY = INT
( RND (1) * 7 + 1)
6890 IF XY = 1 THEN PRINT "BAH!  AN EXI
T TO THE NORTH IS WORTH AT LEAST ";CO
6900 IF XY = 2 THEN PRINT "NO, DO I LOO
K LIKE THE KIND OF MAN"
6905 IF XY = 2 THEN PRINT "WHO WOULD AC
CEPT ";Y;"?"
6910 IF XY = 3 THEN PRINT "BAH!  CHICKE
N FEED."
6920 IF XY = 4 THEN PRINT "NO.  I WANT
```



```
" ;CO;" AT THE VERY LEAST."
6930 IF XY = 5 THEN PRINT CO;" MIGHT BE
    REASONABLE, BUT NOT ";Y
6940 IF XY = 6 THEN PRINT "NO. MAKE ME
    A REASONABLE OFFER."
6950 IF XY=7 THEN PRINT"NEVER WOULD I SE
    LL AN EXIT TO THE NORTH FOR A MERE ";Y
6960 CO = CO - INT ( RND (1) * 10 + 1):
    IF CO < X THEN CO = X
6970 GOTO 6780
7000 PRINT : PRINT " ", "BOO!": PRINT
7010 PRINT "THIS ROOM IS HAUNTED BY A GH
    OST!"
7020 IF GN < 1 THEN 7080
7030 M$="":INPUT "YOUR MOVE";M$:M$ = LE
    FT$ (M$,1)
7040 M = M + 1: IF M$ = "X" THEN 4000
7050 IF (M$ = "N") OR (M$ = "S") OR (M$
    = "E") OR (M$ = "W") THEN 7070
7060 PRINT "WHAT???": GOTO 7030
7070 PRINT "THE GHOST WILL NOT LET YOU G
    O THAT WAY.": GOTO 7030
7080 INPUT "DO YOU SHOOT AT THE GHOST";Q
    $:Q$ = LEFT$ (Q$,1)
7090 IF Q$ = "Y" THEN 7110
7100 GOTO 7030
7110 GOSUB 10210
7120 IF X = 0 THEN 7030
7130 PRINT "THE BULLET PASSES HARMLESSLY
    THROUGH THE GHOST."
7140 PRINT "IT'S STILL THERE.": GOTO 708
    0
7150 PRINT "A GORILLA IS IN THIS ROOM!"
7160 IF GN < 1 THEN 7300
7170 IF BN = 0 THEN 7380
7180 X = M + 3
7190 M$="":INPUT "YOUR MOVE";M$:M$ = LE
    FT$ (M$,1)
7200 M = M + 1: IF M$ = "X" THEN 4000
7210 IF (M$ = "N") OR (M$ = "S") OR (M$
    = "W") OR (M$ = "E") THEN 7240
7220 PRINT "WHAT?": IF M = X THEN 7260
```

```
7230 GOTO 7190
7240 PRINT "THE GORILLA WILL NOT LET YOU
GO          THAT WAY."
7250 IF M < X THEN 7190
7260 PRINT "ANNOYED BY YOUR FOOLING AROU
ND, THE    "
7265 PRINT "GORILLA WHUPS YOU UPSIDE THE
HEAD ---- AND THIS MONKEY IS STRONG!"
7280 IF N > 0 THEN 5130
7290 N = M + 50: PRINT "YOU MUST FIND TH
E FIRST AID KIT"
7295 PRINT "WITHIN 50 MOVES!": GOTO 7190
7300 INPUT "DO YOU SHOOT AT THE GORILLA"
:Q$:Q$ = LEFT$(Q$,1)
7310 IF Q$ = "Y" THEN 7330
7320 GOTO 7170
7330 GOSUB 10210
7340 IF X > 5 THEN 7370
7350 PRINT "HA! YOU GOT THE BIG APE!"
7360 G1 = 0: GOTO 1050
7370 GOTO 7260
7380 INPUT "DO YOU TRY FEEDING THE BANAN
AS TO THE  GORILLA":Q$
7390 Q$ = LEFT$(Q$,1): IF Q$ = "Y" THE
N 7410
7400 GOTO 7180
7410 PRINT "THE GORILLA LEAVES, HAPPILY
MUNCHING ON THE BANANAS."
7420 BN = - 1:G1 = 0: GOTO 1050
7430 PRINT "A WICKED WITCH REARRANGES SOM
E OF THE"
7435 PRINT "TRAP DOORS IN THE BUILDING!"
7440 Y=INT(RND(1)*9+1)+1:FOR X=1 TO Y:T(
X) = INT ( RND (1) * 100 + 1): NEXT
7450 GOTO 1060
7460 PRINT " A WICKED WITCH REARRANGES S
OME OF THE  DOORS IN THE BUILDING!"
7480 FOR X=1 TO 75:Y=INT(RND(1)*80+1)+10
:XY=INT(RND(1)*5+1)
7490 IF XY = 1 THEN N(Y) = 0
7500 IF XY = 2 THEN S(Y) = 0
7510 IF XY = 3 THEN E(Y) = 0
```

```

7520 IF XY = 4 THEN W(Y) = 0
7530 Y = INT ( RND (1) * 80 + 1) + 10:X
Y = INT ( RND (1) * 4 + 1)
7540 IF XY = 1 THEN N(Y) = 1
7550 IF XY = 2 THEN S(Y) = 1
7560 IF XY = 3 THEN E(Y) = 1
7570 IF XY = 4 THEN W(Y) = 1
7580 NEXT : GOTO 1070
7600 PRINT "THERE IS A TREASURE CHEST IN
THIS ROOM."
7610 PRINT "BUT IT IS LOCKED.": IF GN <
1 THEN 7650
7620 IF K1 = 0 THEN 7730
7630 GOTO 1080
7650 INPUT "DO YOU TRY TO SHOOT THE LOCK
OFF";Q$:Q$ = LEFT$(Q$,1)
7660 IF Q$ = "Y" THEN 7680
7670 GOTO 7620
7680 GOSUB 10210
7690 IF X < 7 THEN 7620
7700 PRINT "THE TREASURE CHEST IS OPEN!!
":TX = INT ( RND (1) * 888 + 1)
7710 PRINT "IT CONTAINS ";TX;" GOLD COIN
S!"
7720 T5 = - T5:TR = TR + TX:TX = 0: GOT
O 1080
7730 INPUT "DO YOU TRY USING THE KEY";Q$
:Q$ = LEFT$(Q$,1)
7740 IF Q$ = "Y" THEN 7700
7745 GOTO 1080
9999 STOP
10000 LH = 1: FOR X = 1 TO 5:BT(X) = 0:
NEXT
10010 L2 = 0:DF = 50:GF = 0
10020 FOR X=1 TO 45:YX = INT (RND(1) * 8
0 + 1) + 10:N(YX) = 0: NEXT : RETURN
10030 IF M < 10 THEN RETURN
10040 PRINT : PRINT "THERE IS A VENDING
MACHINE IN THIS "
10045 PRINT"ROOM. INSERT 25 COINS FOR F
RESH BATTERIES!"

```

```
10060 Q$="":INPUT "DO YOU USE THE MACHIN
E";Q$:Q$ = LEFT$(Q$,1)
10070 IF Q$ = "Y" THEN 10090
10080 PRINT "OK": RETURN
10090 IF TR < 25 THEN 10120
10100 TR = TR - 25:BT = 125: PRINT "YOU
NOW HAVE A FRESH SET OF BATTERIES."
10110 RETURN
10120 IF TR > 10 THEN 10150
10130 PRINT "FOR TRYING TO CHEAT THE MAC
HINE, YOU ARE SENT TO ROOM #100!"
10140 TR = 0:P = 100: RETURN
10150 TR = 0: PRINT "THE MACHINE EATS AL
L YOUR COINS, BUT  "
10155 PRINT "DOESN'T GIVE YOU ANYTHING B
ECAUSE YOU  DID NOT MEET THE PRICE."
10170 RETURN
10180 W = ABS (V - P): PRINT "VILLAIN I
S ";W;" ROOM";
10190 IF (W = 0) OR (W > 1) THEN PRINT
"S";
10200 PRINT " AWAY FROM YOU.": PRINT : R
ETURN
10209 REM * SHOOT *
10210 IF GN < .1 THEN 10250
10220 PRINT : PRINT "*** BANG! ***": P
RINT
10230 X = INT ( RND (1) * 10 + 1):GN =
GN - .1
10240 RETURN
10250 PRINT : PRINT "*** CLICK ***": P
RINT
10260 X = 0: RETURN
10270 PRINT "THERE IS A MYSTERIOUSLY TIC
KING BOX IN THIS ROOM.": PRINT
10280 INPUT "DO YOU PICK IT UP";Q$
10290 IF Q$ = "Y" THEN BB = 0:BX = M + 1
00
10300 RETURN
10310 PRINT "THERE IS A SAW LYING IN THE
CORNER.": PRINT
```

```
10320 INPUT "DO YOU PICK IT UP";Q$:Q$ =  
LEFT$(Q$,1)  
10330 IF Q$ = "Y" THEN SW = 0  
10340 RETURN  
10350 PRINT "THERE IS A GUN ON THE FLOOR  
HERE.": PRINT  
10360 INPUT "DO YOU PICK IT UP";Q$:Q$ =  
LEFT$(Q$,1)  
10370 IF Q$ = "Y" THEN GN = GN - INT (G  
N)  
10380 RETURN  
10390 PRINT "YOU ARE CARRYING A GUN WITH  
": INT (GN * 10);  
10400 PRINT " BULLET";: IF GN = .1 THEN  
PRINT ".": GOTO 10410  
10405 PRINT "S."  
10410 RETURN  
10420 PRINT "THERE IS A FIRST AID KIT IN  
THIS ROOM."  
10430 IF N > 0 THEN PRINT " ", "YOU ARE  
SAVED!!"  
10440 N = 0: RETURN  
10450 PRINT "THERE IS A MAGIC LAMP CHAIN  
ED TO THE WALL.": PRINT  
10460 INPUT "DO YOU RUB THE LAMP";Q$:Q$  
= LEFT$(Q$,1)  
10470 IF Q$ = "Y" THEN 10490  
10480 RETURN  
10490 PRINT CHR$(147):PRINT : PRINT : PR  
INT "* POOF! *": PRINT  
10495 PRINT "THE LAMP VANISHES.": RETURN  
10500 IF LH = 1 THEN 10520  
10510 PRINT "OH OH. YOU JUST LOST YOUR  
MAP.": LH = 1: RETURN  
10520 PRINT "HEY! YOU JUST FOUND A MAP.  
": LH = 0: RETURN  
10530 PRINT "YOU JUST FOUND A BUNCH OF R  
IPE BANANAS.":PRINT  
10540 INPUT "DO YOU PICK THEM UP";Q$:Q$  
= LEFT$(Q$,1)  
10550 IF Q$ = "Y" THEN BN = 0  
10560 RETURN
```

```
10570 PRINT "YOU ARE CARRYING A BUNCH OF  
BANANAS.": PRINT  
10575 PRINT "DO YOU WANT TO EAT THEM";  
10580 INPUT Q$:Q$ = LEFT$(Q$,1): IF Q$  
= "Y" THEN 10600  
10590 RETURN  
10600 BN = - 1: PRINT : PRINT " ", "BURP  
": PRINT : IF N = 0 THEN RETURN  
10610 N = N + 50: RETURN  
10620 PRINT "THERE IS A BATTERY RECHARGE  
R IN THE ROOM."  
10630 IF BT < 10 THEN BT = 100: GOTO 106  
40  
10635 BT = BT + 50  
10640 RETURN  
10650 PRINT "YOU JUST RAN INTO DAME FORT  
UNE!": TR = TR + 100  
10660 P = INT ( RND (1) * 20 + 1): RETU  
RN  
10670 PRINT "YOU JUST RAN INTO MISS FORT  
UNE!": P = 100  
10680 IF TR > 0 THEN TR = INT (TR / 2)  
10690 RETURN  
10700 PRINT "A GOOD FAIRY PLANTS MORE GO  
LD COINS THROUGHOUT THE BUILDING!"  
10710 PRINT:GF=0:FOR X=1 TO 100:Y=INT(RN  
D(1)*12+1):IF Y>10 THEN Y=0  
10720 GC(X) = GC(X) + Y: NEXT :GC(P) = 0  
: RETURN  
10730 PRINT "THE BATTERIES IN YOUR FLASH  
LIGHT ARE DEAD. ";  
10740 PRINT "YOU CAN NOT SEE WHERE YOU A  
RE GOING.": RETURN  
10750 FOR X = 1 TO 10: IF T(X) = P THEN  
10770  
10760 NEXT : RETURN  
10770 PRINT "YOU JUST FELL THROUGH A TRA  
P DOOR!"  
10780 IF P > 80 THEN 10810  
10790 Y = INT ( RND (150) + 1):XY = P +  
Y: IF XY > 100 THEN 10790  
10800 P = P + Y: GOTO 10760
```

```

10810 P = INT ( RND (1) * 20 + 1) + 80:
      GOTO 10760
10820 IF CR(CC) < 11 THEN PRINT CR(CC);
10830 IF CR(CC) = 11 THEN PRINT "JACK";
10840 IF CR(CC) = 12 THEN PRINT "QUEEN"
      ;
10850 IF CR(CC) = 13 THEN PRINT "KING";
10860 IF CR(CC) = 14 THEN PRINT "ACE";
10870 PRINT " OF ";: IF CS(CC) = 1 THEN
      PRINT "SPADES"
10880 IF CS(CC) = 2 THEN PRINT "DIAMOND
S"
10890 IF CS(CC) = 3 THEN PRINT "CLUBS"
10900 IF CS(CC) = 4 THEN PRINT "HEARTS"
10910 RETURN
10920 PRINT "YOU JUST CAUGHT THE LEPRECH
AUN!"
10930 IF TR < 10 THEN TR = 50: GOTO 1094
0
10935 TR = INT (TR * 2.5)
10940 LQ=INT(RND(1) * 60 + 1) + 30:X =
ABS (LQ - P): IF X < 20 THEN 10940
10950 RETURN
10960 X = ABS (LQ - P): PRINT "THE LEPR
ECHAUN IS ";X;" ROOM";
10970 IF X > 1 THEN PRINT "S AWAY": GOT
O 10980
10975 PRINT "AWAY"
10980 PRINT "BUT THEN HE MOVES"
10990 LQ=LQ+INT(RND(1)*5+1):IF LQ>100 TH
EN LQ=INT(RND(1)*60+1)+30
11000 X=ABS(LQ-P):IF LQ<15 THEN 10990
11010 RETURN

```

Figures 10.1A and 10.1B are a simplified flowchart that illustrates the basics of the game. The routines used in the GREAT ESCAPE program are summarized in Table 10.1. A list of the variables and their meanings is included in Table 10.2.

The primary obstacle in the way of the player's escape is a character known as the villain. The villain tries to capture the escaping hero, and throw him through a trap door to the south-

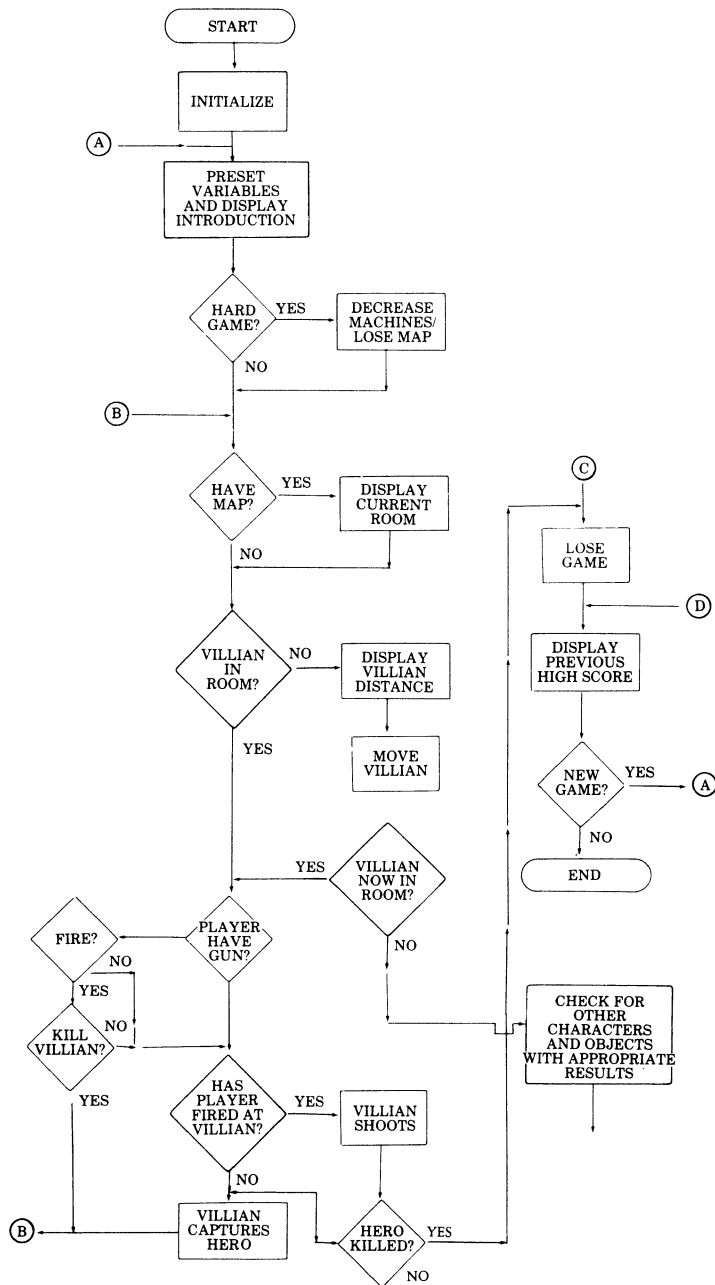


Figure 10.1A Flow-chart for THE GREAT ESCAPE Program.

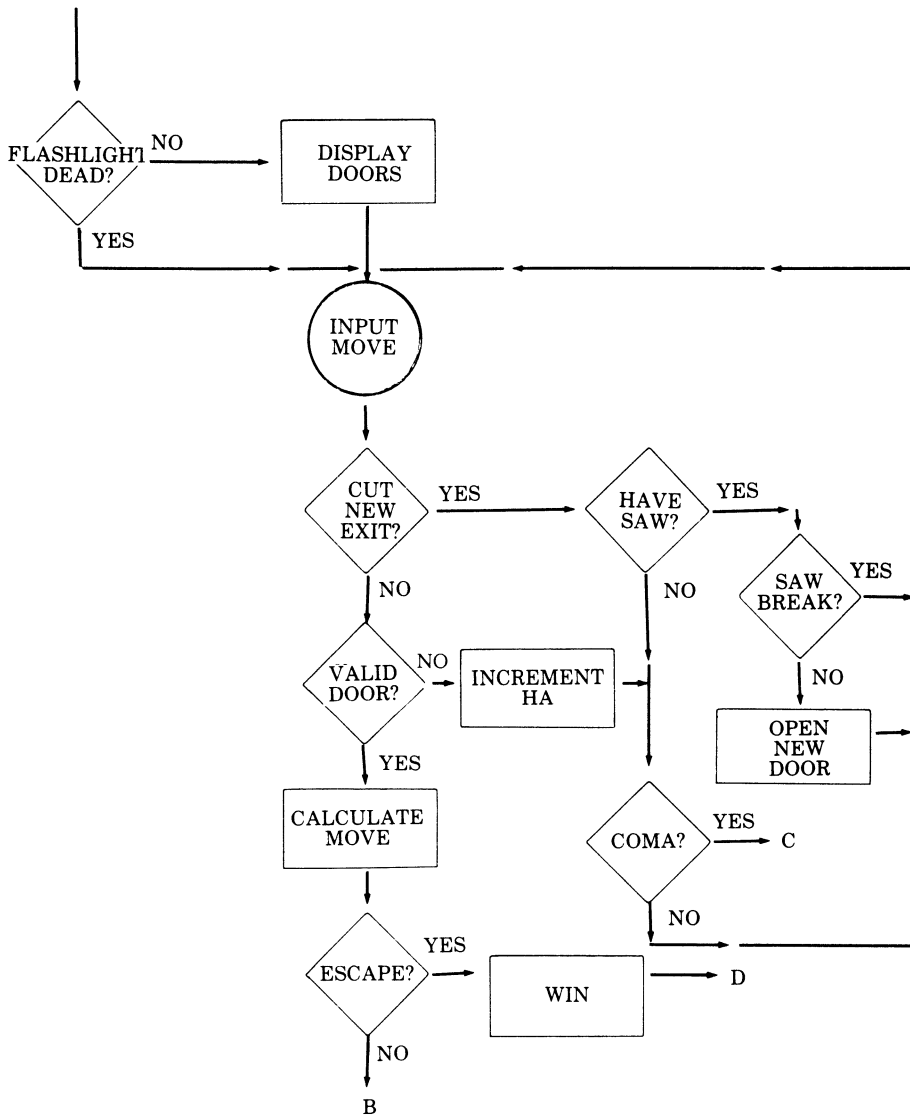


Figure 10.1B Flow-chart for THE GREAT ESCAPE Program.

Table 10.1 Routines and Subroutines Used in THE GREAT ESCAPE Program.**Routines**

10-50	initialize
70-100	shuffle cards
110-450	preset variables and display introduction
500-630	current room display
650-1120	special tests for room occupants
1130-1270	player's move
1280-1330	run into wall
4000-4010	secret passageway
4020	move North
4030	move South
4040	move East
4050	move West
4060-4180	cut new exit
4200-4220	escape — win game
5000-5120	encounter villain
5130-5150	player dead — lose game
5160-6205	display score/end game/begin new game?
5210	kill villain
5220-5240	bomb explodes
5250-5440	Magic Lamp #1
5450-5610	Magic Lamp #2
5620-5750	puppy dog
5760-5790	wounded — must find first aid kit message
5800-5890	snake
5900-6440	robot
6460-6670	mad gambler
6680-6970	evil merchant
7000-7140	Ghost
7150-7420	gorilla
7430-7450	wicked witch rearranges trap doors
7460-7580	wicked witch rearranges building doors
7600-7740	treasure chest

Subroutines

10000-10020	hard game preset
10030-10170	vending machine
10180-10200	villain distance display
10210-10260	shoot
10270-10300	find ticking box

10310-10340	find saw
10350-10380	find gun
10390-10410	carrying gun
10420-10440	find first aid kit
10450-10490	rub Magic Lamp?
10500-10520	find/lose map
10530-10560	find bananas
10570-10610	carrying bananas/eat?
10620-10640	battery recharger
10650-10660	Dame Fortune
10670-10690	Miss Fortune
10700-10720	Good Fairy
10730-10740	batteries dead
10750-10810	trap door
10820-10910	card display
10920-10950	catch leprechaun
10960-11010	leprechaun distance display

Table 10.2 Variables Used in THE GREAT ESCAPE Program.

BB	ticking box location
BN	bananas
BT	battery level
BX	ticking box carried/time to explosion
CC	card counter
CO	evil merchant's counter offer
DF	Dame Fortune
GF	Good Fairy
GH	Ghost
GN	gun
GR	gorilla
G4	game counter
H	previously shot at villain?
HA	run into wall "headache" counter
K	first aid kit
K1	key
LH	have map?
LM	Magic Lamp #1
LQ	Leprechaun
L2	Magic lamp #2
M	move counter
MC	Evil Merchant
MF	Miss Fortune
MG	mad gambler
ML	result of rubbing Magic Lamp

N	wound severity (time to find first aid kit)
O	vending machine present?
P	player's current room location
PD	puppy dog
RB	Robot
RG	Robot's price
RT	game score
RX	previous high score
S	snake
SC	screwdriver
SW	saw
TR	player's collected treasure
TX	coins in treasure chest
T5	treasure chest location
U	miscellaneous
V	villain location
VS	display villain location?
W	villain move/distance to villain
W1	Wicked Witch #1
W2	Wicked Witch #2
X	miscellaneous
X1, X2	player's card
Y, YY	miscellaneous
Y1, Y2	mad gambler's card
Z	miscellaneous

Arrays

BT(12)	vending machines
CR(52)	card values
CS(54)	card suits
E(100)	East doors
GC(100)	gold coins in rooms
N(100)	North doors
S(100)	South doors
T(10)	trap doors
W(100)	West doors

String Variables

M\$	move
Q\$	miscellaneous

most part of the maze building (the farthest from the exits). The villain will also steal all of the player's gold coins.

On each move, the player is informed of the villain's distance before and after he moves. For example:

THE VILLAIN IS 17 ROOMS AWAY!
HE MOVES TO FIND YOU!
THE VILLAIN IS 13 ROOMS AWAY!

At random intervals, the computer will also display the villain's exact location (room number). See line 1110.

The player knows how far away the villain is, but (usually) not which direction. The villain, on the other hand, knows whether the hero is in front of him or behind him, but not the distance. Each time the villain moves, he jumps from 1 to 10 rooms in the correct direction.

Of course, he will often over-shoot and go past the player's position. For example, if the villain is three rooms in front of the hero, he may move back five and end up two rooms behind the hero.

In spite of this, the villain will catch the hero more often than you might expect. When the villain and the player are in the same room the hero is immediately captured, unless he is carrying the gun (which is hidden in one of the rooms at the beginning of the game).

If he has the gun, the player is asked if he wants to shoot the villain (lines 5050 through 5065). The shooting is done in a subroutine that begins at line 10210. Notice that this subroutine checks to make sure the gun has bullets. A random number (X) from 1 to 10 is selected (line 10230) to represent the results of the shot. If this number is greater than 6 (line 5070), the villain is killed (line 5210). For values of X that are six or less, the player's shot misses, and the villain pulls out his own gun and shoots back (lines 5080 through 5120).

The hero has a 40 percent chance of killing the villain, and the villain has a 30 percent chance of killing the hero. Of course, the game ends if the hero is killed.

The villain will remember if the player has shot at him at any time throughout the game. The variable H is used to present this information. If H is set to a value of 1, the villain will shoot at the hero each time they meet, whether the player fires again or not.

If the villain shoots and misses, he will capture the hero in the usual manner.

Incidentally, the villain can move freely from room to room, regardless of the presence of doors. The player's movements are restricted to going through doors.

The villain is not the only obstacle in THE GREAT ESCAPE. A ghost haunts one of the rooms. See lines 7000-7140. This ornery

spirit will block off all regular exists, forcing the player to use a secret passageway, adding 10 to his move count.

A gorilla (lines 7150 through 7420) may also block the movements of the hero. Ordinarily, the only way past the gorilla is to use a secret passageway, as with the ghost. However, if the player has the gun, he may try shooting the gorilla. He should bear in mind that if he misses, he'll have a very upset gorilla on his hands (lines 7260 through 7290).

There is another sure way to get past the gorilla. One of the other rooms in the building contains a bunch of bananas. If the player has found the bananas and picked them up, he can feed them to the gorilla. The grateful ape will then leave and will not reappear for the remainder of the game.

The hero can eat the bananas himself at any time he is carrying them. On each move he will be asked if he wants to eat the bananas. Ordinarily there is no advantage to eating them (other than stopping the constantly repeated question). But if the player is injured and looking for the first aid kit, eating the bananas will extend the time limit.

There is only one bunch of bananas in the maze building. Once the player or gorilla eats them (or they are given to the robot, discussed shortly), they are gone for the rest of the game.

If the player finds himself in the room with the robot (lines 5900 through 6440), a menu of options will be displayed. These include:

CRY

DISMANTLE ROBOT (only if player has screwdriver)

EAT ROBOT

FEED BANANAS TO ROBOT (only if player has bananas)

GIVE COLD COINS TO ROBOT

PUSH PAST ROBOT

SHOOT ROBOT (only if player has gun)

X—USE SECRET PASSAGEWAY

To select the desired action the player enters the first letter. For example, to "Push past robot", enter P. The entire phrase can be typed in, but the program is set up to look at just the first letter.

Several of these options can be helpful, while others have no real effect. Some of them may even be dangerous.

One hint to remember is that every robot has its price but it's very dangerous to offer more than you have.

In his travels through the building, the player may also encounter an evil merchant (lines 6680 through 6970). When this hap-

pens, the player must either go south, or try to bargain with the merchant for passage north. The merchant will make counter offers, but he may accept less. Bargain.

The mad gambler, on the other hand, doesn't give the player any choice in the matter. See lines 6450 through 6670.

When the hero wanders into the mad gambler's room, he must join in a game of High Card. If he wins (draws the high card), his gold coins are doubled, and he moves one room north. If he loses, his gold coins are reduced by half, and his move count increases by three.

The game continues until the player either wins, or loses often enough so that his supply of gold coins becomes too small to merit the mad gambler's interest. If the player does not have enough gold coins to play, he is thrown into room 100.

Two wicked witches also lurk within the maze building. One will rearrange several of the room doors (lines 7460 through 7580), and the other will shift the trap doors to different rooms (lines 7430 through 7450). This may work out to the player's advantage, but it is more likely to be a nuisance.

Some of the rooms are booby-trapped with trap doors, which will randomly relocate the player into the southern half of the maze. This is done in lines 10750 through 10810.

If the player happens to be in the southern most section of the building when he falls through a trap door, he may find himself moved to the north. So trap doors aren't always bad, just usually.

A snake is slithering about in one of the rooms (lines 5800 through 5890). If the player comes into this room, he will be bitten and must locate the first aid kit within 30 moves or die (lose the game).

If he happens to come back into the room with the snake and gets bitten again, he dies immediately.

If the hero is carrying the gun, he may shoot at the snake. Even if he doesn't kill it, he may frighten it off. It will reappear in a nearby room.

Another room contains a puppy dog (lines 5620 through 5750). Usually, the puppy dog will not bother the player, but it will occasionally bite. The hero must then find the first aid kit within 100 moves.

If the player has the gun, the puppy dog is more likely to bite. He will be given the option to shoot at the puppy dog.

Dame Fortune, and her daughter Miss Fortune are also passing through the maze. If the player encounters Dame Fortune (lines 10650 and 10660), his store of gold coins increases.

Running into Miss Fortune (lines 10670 through 10690), isn't quite so pleasurable. The player loses all his gold coins and is placed in room 100.

There is also a mysteriously ticking box in one of the rooms (lines 10270 through 10300). The player is asked if he wants to pick it up. Once he picks up the box, he cannot put it down.

BATTERIES AND MAP

If the player has not escaped from the building within a reasonable number of moves, the batteries in his flash light might go dead. The doors in each room will then not be displayed and the player must blindly try each direction. If there is no door, he will run into a wall. Bumping into too many walls can be quite hazardous to his health.

The player may purchase fresh batteries from one of the vending machines scattered throughout the maze. A set of batteries cost 25 gold coins. A player is advised not to use one of these vending machines when he has fewer than 25 coins.

A battery recharger, hidden in one of the rooms, works automatically at no cost.

The player's map tells him which room he is in, but if he does not have a map, no room number will be displayed.

At the beginning of the game, the computer asks the player if he wants a hard game. If he answers YES, he starts out without a map. There will be fewer vending machines in the building.

A player can find or lose a map randomly throughout the game. Without a map, the player must figure out his location on his own.

AIDS FOR THE PLAYER

Obstacles are a necessary part of any good adventure game. However, don't stack the odds too heavily against the player. The GREAT ESCAPE includes a few helpful items and characters, such as Dame Fortune, hidden within the maze building. As you have seen, some of the obstacles may turn out to be beneficial. For instance, a wicked witch may open a door for you, or move a troublesome trap door. There are a few other helpful things the player might encounter as he attempts to escape:

A saw is hidden in one of the rooms (lines 10310 through 10340). Once the player has a saw, he can cut new exits from the rooms (see lines 4060 through 4180). For instance, if he is in a room with a door to the east, and one to the south, he can use the saw to create a new exit to the north. The new exit will remain in the room for the rest of the game (unless it is moved by a wicked witch).

When a new exit is cut, a corresponding door will not appear in the adjoining room. Ordinarily, if you pass through a door to the north, the new room will have an exit to the south. This will not be the case when a player cuts an exit for himself. The tampering of the wicked witch can also create some one-way doors.

Having the saw would make escape too easy—just cutting exists to the north until you get out of the building. However, the walls are tough, and the saw might break at any time. See lines 4090 and 4150 through 4180. Once the saw is broken, it is gone for the rest of the game. This encourages the player to reserve the saw for when it is really needed.

The saw cannot be used to escape from opposing characters like the ghost, gorilla, evil merchant, or mad gambler.

Also hidden within the building are two Magic Lamps. They will not both appear in the same room.

When he finds one, the player is asked if he wants to rub it. If he says NO, nothing happens, and the game continues normally. Rubbing a Magic Lamp causes one of several things to happen. Usually, the result will be to the player's advantage. The Magic Lamp may kill the gorilla, reduce the player's move count, or move the hero to the room with the saw (among other possibilities).

However, sometimes the Magic Lamp might do something disadvantageous. For example, if the player has the saw and has reached room 13, there is a very good chance that he can escape in a couple of moves. In this case, he probably wouldn't want to be moved to room 87 just because that's where the first aid kit happens to be. Moreover, a Magic Lamp might revive a dead villain, kill Dame Fortune, or even the hero himself. To rub a Magic Lamp is, therefore, a bit of a gamble.

A leprechaun is wandering through the maze. He is extremely difficult to catch, but if the player lands in the same room as the leprechaun, his treasure of gold coins will be increased (lines 10920 through 10950).

The distance to the leprechaun before and after he moves is displayed (lines 10960 through 11010) similarly to the villain. The

leprechaun starts at a random location and moves towards room 100. When he reaches room 100 he jumps back to a new random location and starts over.

A good fairy is hidden in one of the rooms in the maze building. If the player happens to enter this room, the good fairy plants more gold coins throughout the various rooms of the building and vanishes (lines 10700 through 10720). The good fairy will appear only once in the course of a game.

Another room magically doubles the player's hoard of gold coins. Entering this room a second time will have no special effect. The magic only works once.

A treasure chest with a random number of coins (up to 888) is hidden in one of the rooms. See lines 7600 through 7740.

If the player has the gun, he is given the option of trying to shoot the lock off. This has a slim chance of working. Usually, it only wastes a bullet.

If the hero is carrying the key (hidden in another room), he is asked if he wants to try it on the chest's lock.

If the player has neither the gun nor the key, contents of the treasure chest are inaccessible. Of course, the treasure chest may be opened only once per game.

SUMMARY

These are the major points of the game. Many other details of the program are even more fun when they come as a surprise, so they will not be discussed here.

Chapter 11

Marketing Your Software

After all the hard work you've done to create an adventure game, why not let other people enjoy it too? You may be able to make a profit on your time investment.

Good game programs are always in demand, and there are a number of ways to market the software you've written. There are dozens of computer magazines published these days. Most of these regularly publish programs, or would if they could get good ones. These publications need to fill their pages, and they depend on freelance material.

Write to the editor first, rather than send in unsolicited work. Some magazines are reluctant to look at unsolicited submissions, especially from people they don't know.

Keep your letter short; a single typed page. Always type your query letter. Introduce yourself and *briefly* describe your program. If the editor likes the idea, he'll give you a go-ahead. This request is not a firm commitment to buy, or publish, the program in most cases. It means only that the editor is willing to consider it.

Don't be afraid to send in such a letter. The worst an editor will do is send you a form letter rejecting your idea.

If your idea is rejected, don't take it personally. There are many reasons that may have nothing to do with your ability, or the quality of the idea. The editor may have just purchased a similar piece, or he may be over-stocked with game programs, or he might feel that your program just isn't quite right for his particular magazine. If you get a rejection slip shrug and try again.

It's always a very good idea to study back issues of the magazine before making a submission. A publication slanted heavily towards business programs probably wouldn't be interested in an

adventure game, no matter how good it is. Some computer magazines don't publish programs at all.

You should also make sure your program is suitable to the machine(s) covered by the magazine. For instance, *Commander* concentrates exclusively on the Commodore series of microcomputers. They're not going to be very interested in a program written for the TRS-80 unless you can adapt it for a Commodore computer.

Book publishers are another possibility, especially if you have written several related programs. However, the market has been somewhat swamped with program books. These days a book of simple programs can be rather difficult to sell unless it has an unique slant.

A third potential market for your programs is software publishers, a few of which are listed with their addresses in Figure 11.1. Check the ads in computer magazines for more names and addresses.

Some companies sell stock programs, usually on cassette tapes or floppy discs. Most software publishers work on a royalty basis. That is, the more copies of your program they are able to sell, the more money you will be paid. Typical contracts give you 10 to 20 percent of the selling price for the program. Often several programs will be placed on a single tape, or disc. In this case, the royalty money will be split up between the authors of each program.

Find a publisher that deals with programs for your brand of computer, and write them a brief letter describing your program(s), and asking what submission format they prefer. If they are interested, they will give you instructions.

Some software publishers are not particularly interested in submissions by amateurs. If you are not sure, query anyway. They can only say "No".

Generally, you will be asked to supply some kind of documentation, or instruction, booklet for your program. While this doesn't have to be a masterpiece of literature, it is vitally important. Take your time, and make it as good as you can. Make sure you've covered everything the user needs to know thoroughly and clearly. You won't be standing at the customer's side to give instructions.

Some independent programmers take out ads in the computer magazines and try to sell their programs directly themselves.

While this has proven quite profitable in a number of cases, more often it results in a major disaster. For one thing, the ads are expensive, and you have to sell quite a few copies to make the enterprise worthwhile.

Unless you are familiar with the intricacies of setting up a small business and dealing with mail order, it is probably best to avoid this approach.

Computer games are extremely popular these days, and they show every sign of retaining their popularity in the near future. If you've written a good game program, you may be able to use it as a nice supplement to your income. While you probably won't be able to make a living at it, it can be a very profitable hobby.

Have fun.

Adventure International
507 East Street
P.O. Box 3435
Longwood, FL 32750

Arcsoft Publishers
P.O. Box 132
Woodsboro, MD 21798

Broderbund Software Inc.
Entertainment Software Division
1938 4th Street
San Rafael, CA 94901

Datamost
9748 Cozycroft Avenue
Chatsworth, CA 91311

EPYX
P.O. Box 4247
Mountain View, CA 94040

Instant Software
Peterborough, NH 03458

K-Byte
1705 Austin Street
Troy, MI 48084

Quality Software
Suite 105
6660 Reseda Boulevard
Reseda, CA 91335

Sirius Software Inc.
10364 Rockingham Drive
Sacramento, CA 95827

Figure 11.1 Selected Software Publishers.

Loading Instructions

What You Need to Run Adventure Games for the Commodore 64 computer:

- Commodore 64 computer
- One disk drive or datassette cassette recorder
- Standard monitor or TV

Preliminary Steps

The instructions for loading and running the programs on your Commodore 64 computer are divided into two sections: Section 1 covers a Commodore 64 with a disk drive; and Section 2 covers a Commodore 64 computer with cassette recorder.

The Overall Plan

For the Commodore 64, you should have one disk or cassette with the program on it. The book provides program descriptions, and instructions for running and using the programs. Refer to your owner's manual for the proper way to insert a diskette. NEVER remove a diskette from the drive while the red light is still on, as this may cause damage to the diskette.

NOTE: It is a good idea to back up your program diskette. See your owner's manual for instructions on How to Create a Backup Disk.

SECTION 1: COMMODORE 64 WITH A DISK DRIVE

Start Up Steps

To load and run the programs, follow these steps:

- A. Insert the Golden Flutes and Great Escapes program disk into your disk drive (Drive 1 if you have more than one disk drive), and turn on the computer. Type LOAD "GOLDEN FLUTES AND GREAT ESCAPES", 8. The red light on the disk drive will light up, and the drive will make some whirring noises.

The screen displays:

```
SEARCHING FOR GOLDEN FLUTES AND  
GREAT ESCAPES  
LOADING  
READY
```

- B. When the READY message appears and the cursor can be seen below it, just type RUN and press the RETURN key.
- C. Then a menu screen with your selection of adventure games will appear, along with the dilithium Press copyright information.
- D. The menu options are as follows:
 - 1) THE GOLDEN FLUTE
 - 2) THE GREAT ESCAPE
 - 3) TREASURE HUNT
 - 4) MARS
 - 5) QUIT
- E. You are ready to load a program. Simply press the number of your choice, and the program is ready to go. If at any time you want to stop a program, press the RUN/STOP (located in the lower left corner of the keyboard). This takes you back to the menu screen, where you can either make another selection or quit playing.

SECTION 2: COMMODORE 64 WITH A DATASSETTE CASSETTE RECORDER

Preliminary Steps

To make things easy for yourself in the future, you should make a list of the programs and their tape counter positions. Make sure

that the tape is rewound, then set the tape counter to zero, type **LOAD** and press the **RETURN** key.

The computer will respond with a message telling you to press **PLAY** on the cassette recorder. When you press **PLAY**, the computer will load the first program it finds on the tape, and will display a **READY** message when it's through loading.

On a piece of paper, write down the name of the program it finds and mark zero as its tape counter position. Then note the new tape counter setting at the beginning of the next program. Do this for each program on both sides of the tape. Note: don't forget to reset the counter to zero before starting a different side of the cassette.

Start Up Steps

- A. Turn on the computer and TV set (or monitor).
- B. With the cassette rewound to the beginning of the tape on either side, type in the following command:

LOAD "MENU"

and press the **RETURN** key. The computer will respond with: **PRESS PLAY ON TAPE**. You should do this. The screen will go blank as the computer searches for the program called **MENU**. When found, the screen will come back with several messages:

OK
SEARCHING FOR MENU
FOUND MENU

If you want to continue to load the program, you can wait about 10 seconds and the computer will automatically load it. If not, you can speed up the process by pressing the **C =** key, which is located in the lower left corner of the keyboard. This will tell the computer to load the program. If you don't want to load the program, press the **RUN/STOP** key, which is above the **C =** key.

If you continue loading, the screen will go blank until the program is loaded into memory.

- C. You can now type **RUN**, press **RETURN**, and the program will start. The menu program will display everything that is contained on this side of the cassette tape.

DIRECTORY

Side A

1. The Golden Flute
2. The Great Escape

Side B

1. Treasure Hunt
2. Mars

- D. When you choose a game, type the number that appears next to the program you would like to run and press RETURN. The computer will respond with:

POSITION TAPE, THEN PRESS RETURN

At this point, you should refer to your list of tape counter positions (see PRELIMINARY STEPS section), and find the correct number for the program you chose. Forward the tape to this location, and then press RETURN. The computer will display the LOAD command for you, followed by the program you selected. Then all you need to do is press RETURN. The load process will continue as before:

PRESS PLAY ON TAPE
OK
SEARCHING FOR PROGRAM
FOUND PROGRAM
LOADING
READY

- Now type RUN and press RETURN. The program will begin.
- E. You don't need to load MENU each time you want to run a program. The menu has mainly been provided as a list of program names, and is not necessary if you already know which one you want to play. In this case, simply type LOAD "(program name)".

* There is no warranty or representation by dilithium Press or the authors that these programs will enable the user to achieve any particular result.

©dilithium Press 1984

Suite 151

8285 S.W. Nimbus Ave.

Beaverton, OR 97005-6401

(503) 646-2713 or

toll free (800) 547-1842

Index

- Adding commands, 27
- Adventure International, 265
- Aids, 259
- Arcsoft Publishers, 265

- BASIC, 3, 159
- Batteries, 259
- BLAST OFF, 52, 58, 80
- Brinchley Beast, 66
- Broderbund Software, 265

- Cassette recorder, 268
- Characters, 216
- CHR9(\$), 225
- CLIMB, 96
- Commander**, 264
- Commodore 64, ix, 3, 157, 267
- CRY, 28, 34, 58, 84, 257

- Datamost, 265
- Dead monsters, 80
- DIA, 58
- DIAGNOSIS, 28, 32
- DIM, 11
- Disk drive, 268
- DRINK, 53, 58, 90
- DROP, 41, 58

- EAT, 53, 58, 86, 257
- Encountering obstacles, 185

- FILL, 92
- Finding objects, 37
- Flow charts, 13, 23, 29, 35, 41, 54, 61, 65, 78, 181, 211, 251
- FOR . . . NEXT, 18, 28, 38
- Funny-colored sky, 77

- Game format, 6
- GET, 28, 41, 58, 120, 162, 225
- Ghost, 63
- GIVE, 257
- GOSUB, 19, 33
- Graphics, 158
- Grimph, 69

- HELP, 34, 58

- IF . . . THEN, 18, 25, 27, 33, 49, 53, 80, 218, 225
- INFLATE, 95
- Initialization, 11
- INPUT, 19, 120, 162, 183
- Instructions, 119
- INV, 50, 58
- INVENTORY, 50

- KILL, 104
- Kufu, 67

- Loading instructions, 267
- LOOK, 34, 58

- Main play routine, 21
- Mapping, 59, 218, 259
- MARS, 3, 5-156
- Marsquake, 73
- Monsters, 9, 81, 218, 223
- Mountain/Ravine, 72, 84

- Naming, 8

- Obstacles, 10, 228
- ON . . . GOTO, 27, 33
- OPEN BOX, 100, 104

- POKE, 159
- PRAY, 34, 58, 102
- PRINT, 17, 27, 33, 81, 120, 225
- Program listings, 125, 167, 191, 229
- Purofolee, 70

- Random, 219
- Real-time inputs, 162
- REM, 11, 34, 225
- River, 71, 84
- RND, 219
- Routines, 116, 176, 209, 253

- SAVE GAME, 163
- SCORE, 28, 31, 58
- Scoring, 227
- Setting values, 14
- Sound effects, 161
- Squeanly Serpent, 62
- Start-up, 268
- STOP, 19
- Storm, 75
- Subroutines, 117, 176, 210, 253

- THE GOLDEN FLUTE, 3, 189
- THE GREAT ESCAPE, 3, 227
- TOUCH, 114
- TREASURE HUNT, 3, 165

- Variables, 117, 177, 215, 254

- WAIT, 34, 58
- Weapons, 112

- CITY, STATE, ZIP _____

[illegible]

- CITY, STATE, ZIP _____

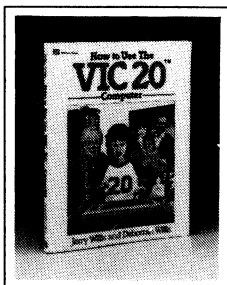
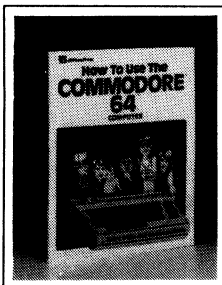
800-547-1842

- CITY, STATE, ZIP _____

[illegible]

MORE HELPFUL WORDS FOR YOU

from  dilithium Press



HOW TO USE THE COMMODORE® 64™

Jerry Willis and Deborah Willis

ISBN 0-88056-133-5

144 pages/20 illustrations

\$5.95

HOW TO USE THE VIC 20™ COMPUTER

Jerry Willis and Deborah Willis

ISBN 0-88056-134-3

184 pages/32 illustrations

\$5.95

A helpful introduction to your computer and its basic components, this book shows you how to set up or install your computer and get it up and running in no time! Learn how to load and save your own programs as well as programs published in books and magazines.

KEEPTRAK™

File Manager
for Personal Computers
Norm Church

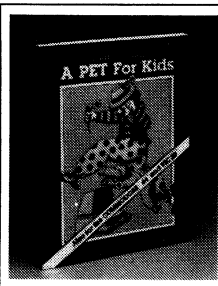
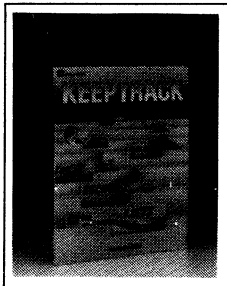
This book and software package turns your Commodore 64 or VIC 20 computer into a personal filing cabinet! It's designed for easy access and will "keep track" of everything from birth-days to tax expenses.

BOOK/SOFTWARE PACKAGES:

ISBN 0-88056-185-8 5 1/4" disk \$29.95

ISBN 0-88056-182-0 cassette \$29.95

Software (either disk or cassette) contains programs for Commodore 64 computers on one side, and programs for VIC 20 computers on the other.



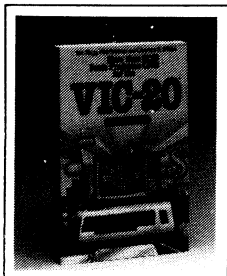
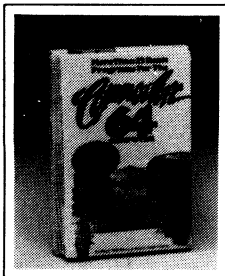
A PET FOR KIDS

Examples work on Commodore 64 and VIC 20 computers.
Sharon Boren

Here is a fun and instructive approach to teaching kids programming in BASIC and computer operation! Illustrations and examples motivate children ages 8 to 13 to higher learning levels.

ISBN 0-88056-106-8 200 pages \$8.95
140 illustrations

Student workbook and teacher's guide available.



MORE THAN 32 BASIC PROGRAMS FOR THE COMMODORE® 64™

Tom Rugg, Phil Feldman and Western Systems Group

BOOK:

ISBN 0-88056-112-2

354 pages/134 illustrations \$19.95

BOOK/SOFTWARE PACKAGES:

ISBN 0-88056-180-7

5 1/4" disk \$39.95

ISBN 0-88056-183-1

cassette \$39.95

MORE THAN 32 BASIC PROGRAMS FOR THE VIC 20™ COMPUTER

Tom Rugg, Phil Feldman and Clarence S. Wilson

BOOK:

ISBN 0-88056-181-5

354 pages \$19.95

BOOK/SOFTWARE PACKAGE:

ISBN 0-88056-058-2

cassette \$39.95

Here is a collection of programs for your entire family. It's chock-full of programs with practical applications, educational uses, games, and graphics too! Type in programs from the book or use ready-to-run programs provided in the book/software package.

BRAIN FOOD — Our free catalog listing over 210 microcomputer books covering software, hardware, business applications, general computer literacy and programming languages.



dilithium Press books are available at your local bookstore or computer store. If there is not a bookstore or computer store in your area, charge your order on VISA or MC by calling our toll-free number, (800) 547-1842.

Send to: dilithium Press, P.O. Box E, Beaverton, OR 97075. Please send me the book(s) I have checked. I understand that if I'm not satisfied I can return the book(s) within 10 days for full and prompt refund.

☐ **HOW TO USE THE COMMODORE 64**

☐ **HOW TO USE THE VIC 20**

KEEPTRAK

Book/Software Packages:

☐ Disk: ☐ Cassette

A PET FOR KIDS

☐ Book ☐ Workbook ☐ Teacher's Guide

MORE THAN 32 BASIC PROGRAMS FOR THE

VIC 20

☐ Book

Book/Software Packages:

☐ Disk ☐ Cassette

MORE THAN 32 BASIC PROGRAMS FOR THE

COMMODORE 64

☐ Book

Book/Software Packages:

☐ Disk ☐ Cassette

☐ Check enclosed \$ _____, payable to dilithium Press

Please charge my Visa ☐ MC ☐

_____ Exp. Date _____

Signature _____

☐ Send me your free catalog, **BRAIN FOOD**.

Name _____

Address _____

City, State, Zip _____

prices are subject to change

MORE HELPFUL WORDS FOR YOU from **dilithium Press**



Bits, Bytes and Buzzwords

Mark Gaetz

This book translates all the computer jargon you've been hearing into words you can understand. It explains microcomputers, software and peripherals in a way that makes sense, so your buying decisions are easier and smarter.

ISBN 0-88056-111-4

160 pages

\$7.95



COMPUTERS FOR EVERYBODY™, 3RD EDITION

Jerry Willis and Merl Miller

In a clear, understandable way, this new edition explains how a computer can be used in your home, office or at school. If you're anxious to buy a computer, use one, or just want to find out about them, read this book first!

ISBN 0-88056-131-9

368 pages

\$9.95



COMPUTERS FOR EVERYBODY™ 1984 BUYER'S GUIDE

Jerry Willis and Merl Miller

Here's a single source for up-to-date information on microcomputers. This book tells you everything you need to know about computer and software buying, including in-depth comparisons of 143 computer models.

ISBN 0-88056-132-7

592 pages

paper \$19.95



Instant (Freeze-Dried Computer Programming in) BASIC—2nd Astounding! Edition

Jerald R. Brown

Here is an active, easy, painless way to learn BASIC. This primer and workbook gives you a fast, working familiarity with the real basics of BASIC. It is one of the smoothest and best-tested instructional sequences going!

ISBN 0-918398-57-6

200 pages

\$12.95



BRAIN FOOD — Our catalog listing of over 210 microcomputer books covering software, hardware, business applications, general computer literacy and programming languages.

dilithium Press books are available at your local bookstore or computer store. If there is not a bookstore or computer store in your area, charge your order on VISA or MC by calling our toll-free number, (800) 547-1842.

Send to: **dilithium Press, P.O. Box E, Beaverton, OR 97075**

Please send me the book(s) I have checked. I understand that if I'm not fully satisfied I can return the book(s) within 10 days for full and prompt refund.

..... Bits, Bytes, and Buzzwords \$7.95

..... Computers For Everybody, 3rd Edition \$9.95

☐ Check enclosed \$
Payable to dilithium Press

..... Computers For Everybody 1984 Buyer's Guide \$19.95

..... Instant (Freeze-Dried Computer Programming in)

BASIC — 2nd Astounding! Edition \$12.95

☐ Please charge my
VISA MASTERCHARGE

_____ Exp. Date _____

Name _____

Address _____

City, State, Zip _____

Signature _____

☐ Send me your catalog, Brain Food.

COMMODORE 64 OWNERS TAKE NOTE . . .

If you enjoyed **GOLDEN FLUTES AND GREAT ESCAPES for the Commodore® 64™**, you'll appreciate having the ready-to-run adventure games on a disk or cassette for your Commodore 64 computer. The software comes with a warranty (if the product is found defective, we will replace it free of charge), as well as a "forever guarantee" (in case of damage, simply return the disk or cassette with \$5 and we'll send you a new one).

Interested?

- ☐ You bet I'm interested in software for **GOLDEN FLUTES AND GREAT ESCAPES for the Commodore® 64™**!

Please send me a

- ☐ 5¼" disk (0-0031-1015A) for my Commodore 64 \$24.95
☐ cassette (0-0031-1015B) for my Commodore 64 \$24.95
☐ Please find my check in the amount of \$25.95 (which includes \$1.00 for shipping) and rush my Commodore 64 software to the address below.
☐ Please charge my VISA _____ M/C _____ \$25.95 (which includes \$1.00 for shipping) and rush my Commodore 64 software to the address below.

Acct# _____ Exp. Date _____

Signature _____

Name _____

Address _____

City, State, Zip _____

(To expedite your order, phone 1 (800) 547-1842 and charge to your VISA or M/C)

- ☐ Please send me your free catalog entitled **BRAIN FOOD™**
dilithium Press books and book/software packages are also available at your local bookstore and computer store.

**PLACE
STAMP
HERE**

dilithium Software

**P.O. Box 606
Beaverton, OR 97075**

You move cautiously down a long, dark corridor beneath the castle of Nembuzur. You notice that your torch is beginning to burn a little low. You'll have to find a new one soon. At the end of the hall you come to three closed doors. After a moment's hesitation, you open the third door. A saber-toothed tiger leaps out at you and gashes your left arm. Quickly you draw your magic sword . . .

Have you ever designed an exciting adventure game in your mind? Would you like to learn how to program it on your Commodore 64 computer? In this book you will learn:

- The basic design rules for original, interactive adventure games.
- The process of how to create your own adventure programs.
- Special programming tricks in BASIC.

Here are four complete adventure game programs with full explanations of how everything works! Programs include Golden Flutes, Great Escapes, Mars, and Treasure Hunt!

You can either type in the programs yourself or buy the book and software as a package. Book/Software packages include a 5¼" disk or cassette containing all of the programs plus example programs, the loading instructions, and a warranty card with our "forever replacement guarantee."

Software is available for the **Commodore 64 computer** with 64K RAM, 1 disk drive or cassette recorder, and monitor or television.

dilithium Press offers technical support over the telephone. With our toll-free number and friendly staff, we can answer all your questions about **dilithium Press** software. Outside of Oregon dial **(800) 547-1842**, in Oregon call 646-2713.



ISBN 0-88056-051-7



dilithium Press
SOFTWARE



—dilithium Press—



Software

WARRANTY REGISTRATION



IMPORTANT!

Mail the enclosed registration
card immediately!

P.O. Box 606, Beaverton, OR 97075
(800) 547-1842 or (503) 646-2713

WARRANTY REGISTRATION CARD

Name _____
Address _____
City _____ State _____ Zip _____
Telephone number _____

Name of program _____
Version number (if applicable) _____
Type of media ☐ Cassette ☐ Diskette
Date purchased _____
Type and model of computer _____

What was this software purchased through?

☐ Bookstore ☐ Brain Food Catalog
☐ Discount store ☐ Magazine ad in _____
☐ Computer show ☐ Other _____

Magazines you subscribe to or read regularly: _____

If this was purchased for a child, what age is the child?

☐ 3-5 ☐ 6-8 ☐ 9-12 ☐ 13-15

Was this program purchased to be used in a classroom setting?

☐ Yes ☐ No

Where is your computer?

☐ Home ☐ Office ☐ School

How many software products do you own? _____

What three products do you use the most? _____

How much computer experience have you had?

☐ Less than 6 months ☐ 1-4 years
☐ 6 months-1 year ☐ 5 years or more

What type of software would you like to purchase in the future? _____

To help us improve our publications, we would appreciate your comments. Please rate:

	GOOD	AVERAGE	POOR
1. Appropriateness of the book/software level to a person of your experience.	☐	☐	☐
2. Organization of book/software.	☐	☐	☐
3. Clarity of information.	☐	☐	☐
4. Completeness of information.	☐	☐	☐
5. Appearance of book/software package.	☐	☐	☐

Did you notice any error in this book and/or software? _____

Comments? _____

LIMITED WARRANTY

Thank you for purchasing this software product. Please take a moment now to read the following information and to fill out and return the enclosed warranty registration card.

WARRANTY

dilithium Press warrants to the original purchaser that the magnetic medium on which this program is recorded is free from defects in materials and faulty workmanship. If this product is discovered to be defective, return it to dilithium Press for replacement.

FOREVER GUARANTEE

If you should damage the magnetic medium, dilithium Press will replace it, providing that the following conditions are met:

1. You have completed and returned the enclosed warranty registration card.
2. You return the damaged medium to dilithium Press along with \$5.00 for shipping and handling.

The warranty and forever guarantee are valid from the date of purchase, and are subject to the following limitations:

1. This software is sold as is, without warranty or representation by dilithium Press or the authors that it will enable the purchaser to achieve any particular result.
2. Neither dilithium Press nor any other person or company shall be liable for indirect, special, or consequential damages that arise from the use of this product.
3. This software product is for the use of the original purchaser only and only on the computer system specified. The program and its documentation may not be copied except for backup purposes and to load the program into the computer in order to execute it.

PLACE
STAMP
HERE

dilithium Software
P.O. Box 606
Beaverton, OR 97075

You move cautiously down a long, dark corridor beneath the castle of Nèmbuzur. You notice that your torch is beginning to burn a little low. You'll have to find a new one soon. At the end of the hall you come to three closed doors. After a moment's hesitation, you open the third door. A saber-toothed tiger leaps out at you and gashes your left arm. Quickly you draw your magic sword . . .

Have you ever designed an exciting adventure game in your mind? Would you like to learn how to program it on your Commodore 64 computer? In this book you will learn:

- The basic design rules for original, interactive adventure games.
- The process of how to create your own adventure programs.
- Special programming tricks in BASIC.

Here are four complete adventure game programs with full explanations of how everything works! Programs include Golden Flutes, Great Escapes, Mars, and Treasure Hunt!

This book/software package includes a 5¼" disk containing all of the programs plus example programs, the loading instructions, and a warranty card with our "forever replacement guarantee."

Software runs on the **Commodore 64 computer** with 64K RAM, one disk drive, and monitor or television.

dilithium Press offers technical support over the telephone. With our toll-free number and friendly staff, we can answer all your questions about **dilithium Press** software. Outside of Oregon dial **(800) 547-1842**, in Oregon call 646-2713.

Golden Flutes & Great Escapes

For the
Commodore 64™

ISBN 0-88056-204-8



Software
Disk
Package

ISBN 0-88056-204-8



45078 02995



dilithium Press
SOFTWARE

